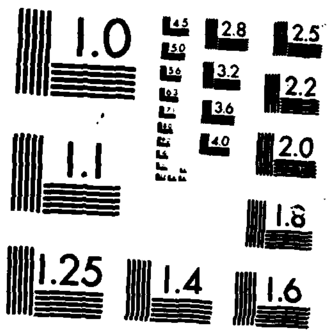
AD-A165 347   ADA (TRADEMARK) TRAINING CURRICULUM INSTRUCTOR'S COURSE   1/2
S500 MANAGEMENT MODULES(U) SOFTECH INC WALTHAM MA 1986
DAAB07-85-C-K506

UNCLASSIFIED   F/G 5/9   NL


1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

AD-A1653[illegible]

1986

Ada® Training Curriculum

Instructor's Course
S500
Management Modules

U.S. Army Communications-Electronics Command
(CECOM)

Contract DAAB07-85-C-K506

Prepared By:

SOFTECH, INC.
460 Totten Pond Road
Waltham, MA 02154

SOFTECH

INSTRUCTOR'S COURSE MODULE (S500)

931/B - MANAGEMENT MODULES

VG 931/B

460 Totten Pond Road, Waltham, MA 02254 (617)890-6900 TWX:710-324-6401

This INSTRUCTOR'S COURSE consists of the following sections:

MANAGEMENT MODULES

(1) OVERVIEW,

(2) ~~E~~101 - Ada ORIENTATION FOR MANAGERS,

(3) ~~M~~101 - SOFTWRE ENGINEERING FOR MANAGERS,

(4) ~~L~~201 - Ada FOR SOFTWARE MANAGERS, and

(5) ~~C~~303 - REAL TIME CONCEPTS. Keywords: ADA programming language, training, instruction manual.

VG 931/B

INSTRUCTOR NOTES

- ALLOW 10 MINUTES FOR THIS SECTION.

SECTION 1

OVERVIEW

INSTRUCTOR NOTES

- REMIND THE INSTRUCTOR'S IN TRAINING THAT COURSE ATTEMPTS TO GIVE THE STUDENTS THE BIG PICTURE. DETAILS ARE PROVIDED IN L201 AND L303.

OVERVIEW OF L101

- GOALS
  - OVERVIEW OF THE DEVELOPMENT OF Ada
  - OVERVIEW OF PROGRAMMING IN Ada
  - OVERVIEW OF Ada FEATURES
  - OVERVIEW OF TRANSITION ISSUES
  - OVERVIEW OF CURRENT STATUS OF THE Ada EFFORT

- GOALS DO NOT INCLUDE
  - TEACH Ada PROGRAMMING

- STUDENT BACKGROUND
  - PROGRAMMING EXPERIENCE (BUT NOT NECESSARILY IN HIGH ORDER LANGUAGES)

- MODULE OVERVIEW (1 DAY)
  - THIS MODULE GIVES AN OVERVIEW OF THE DEVELOPMENT OF Ada, THE Ada LANGUAGE, AND THE Ada ENVIRONMENT. THE COURSE EMPHASIZES THE USE OF Ada IN THE TOTAL PROJECT DEVELOPMENT.

INSTRUCTOR NOTES

- REMIND INSTRUCTOR'S IN TRAINING THAT THEY SHOULD EMPHASIZE SOFTWARE ENGINEERING AND Ada GO HAND-IN-HAND.

OVERVIEW OF M101

- GOALS
  - PROVIDE GENERAL UNDERSTANDING OF SOFTWARE ENGINEERING CONCEPTS
  - ESTABLISH A RELATIONSHIP BETWEEN SOFTWARE ENGINEERING AND Ada
- GOALS DO <u>NOT</u> INCLUDE
  - HOW TO MANAGE SOFTWARE ENGINEERING
- STUDENT BACKGROUND
  - SOME PROGRAMMING EXPERIENCE
  - HOL BACKGROUND <u>NOT</u> ASSUMED
- MODULE OVERVIEW (1 DAY)
  - THIS MODULE DESCRIBES THE SOFTWARE CRISIS, WHAT SOFTWARE ENGINEERING IS, AND HOW IT CAN BE USED TO ALLEVIATE THE SOFTWARE CRISIS. IT ALSO DISCUSSES THE ROLE OF Ada IN SOFTWARE ENGINEERING.

INSTRUCTOR NOTES

- REMIND THE INSTRUCTOR'S IN TRAINING THAT PEOPLE TAKING THIS COURSE ARE INTERESTED IN THE LONG RANGE ASPECTS OF Ada SUCH AS PORTABILITY, REUSABILITY, READABILITY, ETC. THEY ARE NOT INTERESTED IN "WHERE THE SEMICOLONS GO."

OVERVIEW OF L201

- GOALS
  - DEVELOP CONCEPTUAL KNOWLEDGE OF Ada
  - RECOGNIZE HIGH/POOR QUALITY DESIGNS AND CODE IN Ada
  - DEVELOP AN UNDERSTANDING OF PORTABILITY AND REUSABILITY ISSUES

- GOALS DO NOT INCLUDE
  - TEACH Ada DESIGN
  - TEACH Ada CODING

- STUDENT BACKGROUND
  - Ada ORIENTATION FOR MANAGERS (L101) OR Ada TECHNICAL OVERVIEW (L102)
  - SOFTWARE ENGINEERING FOR MANAGERS (M101) OR INTRODUCTION TO SOFTWARE ENGINEERING (M102)

- MODULE OVERVIEW (3 DAYS)
  - THIS MODULE PRESENTS THE ENTIRE Ada LANGUAGE FROM THE POINT OF VIEW OF A TECHNICAL MANAGER WHO WILL DIRECT A SOFTWARE PROJECT WITHOUT PERSONALLY PRODUCING DESIGNS OR CODE

INSTRUCTOR NOTES

- REMIND THE INSTRUCTORS IN TRAINING THAT EXPERIENCE IN REAL TIME OR CONCURRENT PROGRAM IS <u>NOT</u> ASSUMED.

OVERVIEW OF L303

- GOALS
  - PROVIDE CONCEPTUAL UNDERSTANDING OF APPROACHES TO REAL TIME/CONCURRENT PROGRAMMING IN Ada
  - DEMONSTRATE Ada IS A VIABLE LANGUAGE FOR SOLVING REAL TIME PROBLEMS
  - PREPARE PROJECT LEADERS TO UNDERSTAND REAL TIME DESIGNS AND SETTLE DISPUTES
  - UNDERSTAND PERFORMANCE ISSUES

- GOALS DO <u>NOT</u> INCLUDE
  - ENABLE STUDENTS TO WRITE REAL TIME OR CONCURRENT PROGRAMS
  - TEACH SPECIFIC PERFORMANCE IMPROVEMENT TECHNIQUES IN DETAIL

- STUDENT BACKGROUND
  - Ada FOR SOFTWARE MANAGERS (L201)
  - REAL TIME/CONCURRENT PROGRAMMING BACKGROUND <u>NOT</u> ASSUMED

- MODULE OVERVIEW (1 DAY)
  - THIS MODULE PRESENTS THE TASKING FEATURES OF Ada AT THE CONCEPTUAL LEVEL NECESSARY TO UNDERSTAND REAL TIME/CONCURRENT PROGRAMMING ISSUES IN Ada.

INSTRUCTOR NOTES

- ALLOW 30 MINUTES FOR THIS SECTION.

SECTION 2

L101

Ada ORIENTATION FOR MANAGERS

INSTRUCTOR NOTES

GENERAL COMMENTS

- THERE ARE NO IN-CLASS EXERCISES OR LAB EXERCISES FOR THIS MODULE
- SPECIAL CONSIDERATIONS ARE NOTED WHERE APPLICABLE, BASED ON PRIOR EXPERIENCE TEACHING THE MATERIAL
- EACH SECTION IS DISCUSSED, ITS MAIN POINTS AND ITS RELATION TO THE COURSE OVERALL

INSTRUCTOR NOTES

- GIVE THE INSTRUCTORS IN TRAINING AN OVERVIEW OF WHAT IS COVERED IN L101 AND WHAT ITS OBJECTIVES ARE.

- TARGET TEACHING TIME IS IN PARENTHESES
  - INCLUDED TO GIVE INSTRUCTORS IN TRAINING AN IDEA OF HOW MUCH TIME IS DEVOTED TO VARIOUS TOPICS
  - EMPHASIZE THAT TIME IS TARGET. MAY VARY DEPENDING ON CLASS NEEDS/BACKGROUND

OVERVIEW

- SECTION 1 - WHY Ada (:45)
  - DISCUSSES WHY THE Ada EFFORT BEGAN
- SECTION 2 - WHAT Ada IS NOT (:15)
  - CLEARS UP SOME COMMON MISCONCEPTIONS INVOLVING Ada
- SECTION 3 - WHAT Ada IS (3:10)
  - GIVES STUDENTS A FEELING FOR WHAT IT'S LIKE TO PROGRAM IN Ada
  - PRESENT A CATALOG OF Ada FEATURES
- SECTION 4 - WHAT ARE SOME TRANSITION ISSUES WITH Ada? (1:15)
  - DISCUSSES HOW TO TRANSITION TO Ada
  - DISCUSSES HOW TO PROVIDE Ada TRAINING
- SECTION 5 - WHERE IS Ada NOW AND TOMORROW? (:35)
  - DISCUSSES CURRENT STATE OF Ada EFFORT
  - DISCUSSES DoD PLANS FOR TOMORROW

INSTRUCTOR NOTES

- L101 STUDENTS NEED TO UNDERSTAND THAT
  - THERE IS A SOFTWARE CRISIS
  - DoD HAS RESPONDED TO IT (AND STILL IS)
  - THE Ada EFFORT IS THE (ONGOING) RESPONSE
  - Ada WITHOUT SOFTWARE ENGINEERING IS LIKE A MORNING WITHOUT SUNSHINE

SECTION 1 - WHY Ada?

SUMMARY OF MAIN POINTS COVERED:

- ESTABLISH RATIONALE AND HISTORY BEHIND THE Ada EFFORT

MAIN MESSAGES:

- Ada EFFORT SUPPORTS DoD COMMITMENT TO DEVELOPING SOFTWARE THAT IS
  - MAINTAINABLE
  - UNDERSTANDABLE
  - RELIABLE
  - EFFICIENT

SUBTOPICS:

- DoD MOTIVATION
  - SOFTWARE CRISIS
  - REUSABILITY, PORTABILITY
- DoD RESPONSE
  - Ada LANGUAGE
  - Ada SUPPORT ENVIRONMENT
  - SOFTWARE ENGINEERING
- Ada HISTORY

INSTRUCTOR NOTES

SECTION 2 - WHAT Ada IS NOT

SUMMARY OF MAIN POINTS COVERED:

- POSSIBLE MISCONCEPTIONS REGARDING Ada ARE DISPELLED

MAIN MESSAGES:

- Ada IS MORE THAN A PROGRAMMING LANGUAGE
- Ada IS LESS THAN A PANACEA FOR PRODUCTIVITY PROBLEMS AND THE SOFTWARE CRISIS

SUBTOPICS:

- MISCONCEPTIONS
  - Ada WILL SO[illegible] YOUR ORGANIZATION'S PROBLEMS
  - Ada WILL [illegible]ASE PRODUCTIVITY
  - Ada IS J[illegible] ANOTHER PROGRAMMING LANGUAGE

SPECIAL CONSIDERATIONS:

- BEFORE PRESENTING THIS SECTION ASK THE STUDENTS WHAT Ada MEANS TO THEM. THIS BRINGS OUT TYPICAL MISCONCEPTIONS.

INSTRUCTOR NOTES

- REMIND THE INSTRUCTOR'S IN TRAINING THAT THE TITLE FOR THIS COURSE IS <u>Ada ORIENTATION FOR MANAGERS</u>. THE OBJECTIVES OF THIS COURSE DO <u>NOT</u> INCLUDE A COMPREHENSIVE INTRODUCTION TO Ada (THE ROLE OF L201).

- "WHAT IS Ada" DISCUSSES Ada AS A TOOL TO ACHIEVE DoD'S OBJECTIVES, A LANGUAGE SUPPORTING MODERN SOFTWARE ENGINEERING CONCEPTS, AND A COMMON PROGRAMMING SUPPORT ENVIRONMENT.

- "WHAT IS IT LIKE TO PROGRAM IN Ada" GIVES THE L101 STUDENTS A FEELING FOR THE Ada SOFTWARE DEVELOPMENT APPROACH BY USING AN EXAMPLE PROBLEM.

- THE CATALOG OF Ada FEATURES BRIEFLY DESCRIBES THE MAJOR Ada FEATURES, THEIR INTENDED USE, AND (FOR THOSE FEATURES NOT DISCUSSED IN THE EXAMPLE PROBLEM) EXAMPLES OF THEIR USE.

SECTION 3 - WHAT Ada IS

SUMMARY OF MAIN POINTS COVERED:

- WHAT IS Ada?
- WHAT IS IT LIKE TO PROGRAM IN Ada?
- CATALOG OF Ada FEATURES

MAIN MESSAGES:

- THE Ada EFFORT INVOLVES THE Ada LANGUAGE AND AN Ada PROGRAMMING SUPPORT ENVIRONMENT
- Ada IS A READABLE LANGUAGE
- SYSTEM CHANGE CAN BE RELATIVELY EASY IF DESIGNED FOR MODIFIABILITY
- THE LANGUAGE FEATURES SUPPORT THE SOFTWARE ENGINEERING OBJECTIVES OF MAINTAINABILITY, UNDERSTANDABILITY, RELIABILITY AND EFFICIENCY

SUBTOPICS:

- Ada IS A TOOL AND A PROGRAMMING ENVIRONMENT
- HIGH ORDER LANGUAGES; COMPILERS
- PROBLEM REQUIREMENTS
- DECOMPOSITION OF SOLUTION
- Ada IMPLEMENTATION (CODE AND COMPILATION)
- SYSTEM CHANGE

INSTRUCTOR NOTES

SECTION 3 - Continued

SPECIAL CONSIDERATIONS:

- MATERIAL ON HOLS CAN BE SKIPPED IF STUDENTS ALREADY HAVE HOL BACKGROUND
- EXAMPLE PROGRAM IS GIVEN TO SHOW STUDENTS WHAT IT IS LIKE TO PROGRAM IN Ada
  - MANAGERS WHO EXPERIENCE WHAT IT'S LIKE TO PROGRAM IN Ada MAY BE BETTER ABLE TO HANDLE VALID/INVALID PROBLEMS OF THEIR DESIGNERS/PROGRAMMERS
  - PROBLEM PRESENTED THROUGH
    - REQUIREMENTS
    - DESIGN
    - IMPLEMENTATION
  - CONCEPTS STRESSED, SO BE CAREFUL ABOUT SPENDING TIME ANSWERING SYNTAX QUESTIONS
  - DIFFERENT APPROACHES TO COMPILING SYSTEM PROVIDED
    - EMPHASIZE Ada PROVIDES FOR PIECES OF SYSTEM TO BE DEVELOPED IN A MANNER BEST SUITED FOR THE PROJECT
    - IMPACT OF SYSTEM CHANGES CAN BE MINIMIZED THROUGH SEPARATE COMPILATION
  - DO NOT GO INTO DETAIL ABOUT ALGORITHMS IN THE EXAMPLE OR ABOUT THE Ada FEATURES

INSTRUCTOR NOTES

- Ada SOFTWARE DEVELOPMENT PLACES A DIFFERENT EMPHASIS ON LIFE CYCLE EFFORTS
  - LESS TIME IS SPENT ON TESTING
  - MUCH MORE TIME CAN BE SPENT ON DESIGN

- THE PREDICTABLE OBJECTIONS SHOW MANAGERS SOME OBJECTIONS THEY MIGHT HEAR AND WHAT THEY PROBABLY "MEAN".

SECTION 4 - WHAT ARE SOME TRANSITION ISSUES WITH Ada?

SUMMARY OF MAIN POINTS COVERED:

- SOFTWARE DEVELOPMENT IN Ada EMPHASIS ON DIFFERENT PORTIONS OF THE LIFE-CYCLE
- SELECTING AN IMPLEMENTATION/A TRAINING PROGRAM
- NEED FOR A TRANSITION PLAN AND PROPER TRAINING

MAIN MESSAGES:

- DIFFERENCES IN THE Ada SOFTWARE DEVELOPMENT APPROACH ALLOW MODERN SOFTWARE ENGINEERING CONCEPTS TO BE APPLIED MORE NATURALLY
- Ada TRAINING MUST BE PART OF ANY TRANSITION PLAN
- AN Ada PROGRAM DESIGN LANGUAGE (PDL) IS A GOOD WAY TO EASE INTO AN Ada TRANSITION

SUBTOPICS:

- Ada SOFTWARE DEVELOPMENT
    - DESIGN TIME VS. TEST TIME
    - PROBLEM VS. MACHINE
- PREDICTABLE OBJECTIONS A MANAGER MIGHT HEAR
- TEXTBOOK APPROACH TO Ada TRAINING
- A FRAMEWORK FOR Ada TRANSITION

INSTRUCTOR NOTES

SPECIAL CONSIDERATIONS:

- IT MAY COME AS A SURPRISE TO MANY MANAGERS THAT THE TEXTBOOK APPROACH IS NOT ADEQUATE
  - EMPHASIZE THAT THE Ada MODULES IN THIS CURRICULUM TEACH HOW TO USE THE FEATURE EFFECTIVELY, NOT JUST THE FEATURE
  - FOR EXAMPLE, L305 CONTAINS A SECTION ON HOW TO DESIGN WITH EXCEPTIONS
  - FOR EXAMPLE, L401 DISCUSSES TASK DESIGN, NOT JUST HOW TASKS WORK
- AFTER PRESENTING THE FRAMEWORK FOR Ada DESIGNS, CONSIDER ENCOURAGING THE STUDENTS TO EXPRESS THEIR VIEWS OR QUESTIONS ABOUT TRANSITION. ENCOURAGE STUDENTS TO ANSWER THE QUESTIONS ASKED.
- STUDENTS MIGHT ASK IF MORE TIME IS SPENT ON CODING
  - ANSWER IS YES
  - ALLOWING ERRORS TO BE DETECTED DURING COMPILATION RATHER THAN LATER IN THE LIFE CYCLE REQUIRES WRITING DOWN MORE DETAILS
  - READABILITY FOR DESIGN/CODE REVIEWS AND FOR LIFE CYCLE MAINTENANCE IS EMPHASIZED OVER EASE OF WRITING
  - LIFE CYCLE COSTS STILL DECREASED

INSTRUCTOR NOTES

SECTION 5 - WHERE IS Ada NOW AND TOMORROW?

SUMMARY OF MAIN POINTS COVERED:

- OVERVIEW OF ONGOING DoD Ada ACTIVITIES SUCH AS STARS AND STANDARDIZATION. MAKE THE STUDENTS AWARE OF ORGANIZATIONS SUCH AS ADAJUG, ADATEC, AND AJPO.

MAIN MESSAGES:

- DoD IS CONTINUING ITS EFFORTS TO PROVIDE THE Ada COMMUNITY WITH THE BEST IN TOOLS AND METHODOLOGIES
- STANDARDIZATION WILL REMOVE MANY OF THE PROBLEMS CREATED BY DIALECTS
- Ada IS BEING USED
- THERE ARE ORGANIZATIONS THAT CAN PROVIDE YOU WITH Ada INFORMATION

SUBTOPICS:

- STARS
- METHODMAN
- AJPO
- ADAJUG
- ADATEC

INSTRUCTOR NOTES

GO OVER THE OUTLINE QUICKLY TO GIVE A FEEL FOR WHAT THE M101 IS ALL ABOUT

SECTION 3

M101 SOFTWARE ENGINEERING FOR MANAGERS

INSTRUCTOR NOTES

M101 SOFTWARE ENGINEERING FOR MANAGERS

SECTION 1 - BACKGROUND AND MOTIVATION

- DEFINITIONS
- THE SOFTWARE CRISIS
- THE ENVIRONMENT FACING SOFTWARE ENGINEERING

SECTION 2 - SOFTWARE ENGINEERING GOALS

- SUMMARY OF POTENTIAL SOFTWARE ENGINEERING GOALS
- OBJECTIVES
- CONFLICTS BETWEEN GOALS
- SOFTWARE ENGINEERING PRINCIPLES

INSTRUCTOR NOTES

M101 SOFTWARE ENGINEERING FOR MANAGERS - Continued

SECTION 3 - ACHIEVING SOFTWARE ENGINEERING GOALS

- SOFTWARE LIFE CYCLE
- INTRODUCTION TO THE METHODS
- METHODS FOR EACH PHASE OF THE LIFE CYCLE
- SOFTWARE MANAGEMENT METHODS AND TECHNIQUES

SECTION 3A - SOFTWARE LIFE CYCLE

- THE LIFE OF SOFTWARE
- A LIFE CYCLE MODEL (THE TRADITIONAL VIEW)
- SOFTWARE MAINTENANCE ACTIVITIES
- SHORTCOMINGS OF THE TRADITIONAL VIEW

SECTION 3B - INTRODUCTION TO METHODS AND TOOLS

- OUR SCOPE OF CONTROL
- ATTRIBUTES OF METHODOLOGIES
- WHY LEARN METHODOLOGIES
- REQUIREMENTS FOR AN IDEAL METHODOLOGY

SECTION 3C - ANALYSIS OVERVIEW

- DEFINITIONS
- ROLES OF INDIVIDUALS
- CONSEQUENCES OF WRONG REQUIREMENTS

INSTRUCTOR NOTES

M101 SOFTWARE ENGINEERING FOR MANAGERS - Continued

SECTION 3D - ANALYSIS METHODS

- SADT
- SREM
- PSL/PSA
- SSA
- SCRP

SECTION 3E - DESIGN OVERVIEW

- DEFINITIONS
- ROLE OF THE DESIGNER

SECTION 3F - ARCHITECTURAL DESIGN METHODS

- SCRP
- OBJECT ORIENTED DESIGN
- STRUCTURED DESIGN
- JACKSON AND WARNIER ORR
- HIGH ORDER SOFTWARE

SECTION 3G - DETAILED DESIGN METHODS

- PROGRAM DESIGN LANGUAGES
- HIPO
- NSSF

INSTRUCTOR NOTES

M101 SOFTWARE ENGINEERING FOR MANAGERS - Continued

SECTION 3H - IMPLEMENTATION OVERVIEW

- SCOPE OF IMPLEMENTATION PHASE
- IMPLEMENTATION ISSUES

SECTION 3I - IMPLEMENTATION METHODS

- MOTIVATION/PURPOSE OF STRUCTURED DESIGN
- STRUCTURED DESIGN CONCEPTS AND GUIDELINES
- TESTING AS AN ERROR REMOVAL TECHNIQUE
- UNIT TESTING
- INTEGRATION STRATEGIES

SECTION 3J - SOFTWARE MANAGEMENT

- SOFTWARE PLANNING AND TRACKING
- PLANNING TECHNIQUES
- SOFTWARE COST ESTIMATION
- SOFTWARE QUALITY MANAGEMENT
- SOFTWARE CONFIGURATION MANAGEMENT

SECTION 4 - SOFTWARE ENGINEERING AND Ada

- ANALYSIS AND Ada
- DESIGN AND Ada
- STRUCTURED DESIGN AND Ada
- HOW Ada SUPPORTS THE GOALS AND PRINCIPLES

INSTRUCTOR NOTES

MAKE IT CLEAR THAT M101 IS ORIENTED TOWARDS MANAGERS WHO WILL IN MOST CASES NOT HAVE A STRONG COMPUTER ENGINEERING BACKGROUND

M101 IS A SURVEY COURSE AT BEST SO IT IS CRITICAL TO GAIN A CONSENSUS ON THE GOALS

M101 SOFTWARE ENGINEERING FOR MANAGERS

- COURSE GOALS ARE:
  - TO PROVIDE A GENERAL UNDERSTANDING OF SOFTWARE ENGINEERING CONCEPTS
  - TO ESTABLISH A RELATIONSHIP BETWEEN SOFTWARE ENGINEERING AND Ada

- COURSE GOALS ARE NOT:
  - HOW TO MANAGE SOFTWARE ENGINEERING OR THE TRANSITION TO A NEW METHODOLOGY
  - TO MAKE THE STUDENT AN EXPERT IN ANY OF THE TECHNIQUES

INSTRUCTOR NOTES

THE SOFTWARE CRISIS HAS BEEN WRITTEN ABOUT IN HUNDREDS OF ARTICLES SO THE CLASS SHOULD BE ABLE TO RELATE TO IT.

SECTION 1 - BACKGROUND AND MOTIVATION

SUMMARY OF MAIN POINTS COVERED:

- DEFINE SOFTWARE ENGINEERING FROM SEVERAL DIFFERENT POINTS OF VIEW
- MOTIVATION FOR SOFTWARE ENGINEERING IS THE COSTLY NATURE OF SOFTWARE

MAIN MESSAGES:

- THERE IS MORE THAN ONE DEFINITION OF SOFTWARE ENGINEERING
- SOFTWARE IS COSTLY TO DEVELOP AND MAINTAIN

SUBTOPICS:

- DEFINITIONS
- THE SOFTWARE CRISIS
- THE ENVIRONMENT FACING SOFTWARE ENGINEERING

SPECIAL CONSIDERATIONS:

- EMPHASIZE THAT SOFTWARE ENGINEERING IS AN EVOLVING DISCIPLINE
- FOCUS ON THE MAGNITUDE OF THE SOFTWARE CRISIS NOT ON THE DETAILS OF THE NUMBERS PRESENTED
- RELATE ANY RELEVANT PERSONAL EXPERIENCES

INSTRUCTOR NOTES

SECTION 2 - SOFTWARE ENGINEERING GOALS

SUMMARY OF MAIN POINTS COVERED:

- A DISCUSSION OF THE OBJECTIVES AND GOALS OF SOFTWARE ENGINEERING AND THE PRINCIPLES THAT SOFTWARE ENGINEERING IS BUILT ON

MAIN MESSAGES:

- SOFTWARE ENGINEERING GOALS WILL DIFFER BASED ON THE PROJECTS OR ORGANIZATION NEEDS
- WELL DEFINED GOALS AND PRINCIPLES FORM THE FOUNDATION FOR SOFTWARE ENGINEERING

SUBTOPICS:

- SUMMARY OF POTENTIAL SOFTWARE ENGINEERING GOALS
- OBJECTIVES
- CONFLICTS BETWEEN GOALS
- SOFTWARE ENGINEERING PRINCIPLES

SPECIAL CONSIDERATIONS:

- THIS IS A VERY IMPORTANT SECTION, MAKE IT AS CLEAR AS POSSIBLE
- THE PRINCIPLES AND GOALS WILL BE REVISITED IN SECTION 3 AS WE DISCUSS METHODS SO SUMMARIZE THE PRINCIPLES AND GOALS AT THE END OF THE SECTION TO PREPARE THE CLASS FOR WHAT IS TO COME

INSTRUCTOR NOTES

INDICATE THAT AN INTRODUCTORY SECTION IS INCLUDED FOR EACH MAJOR PHASE OF THE LIFE CYCLE.

SECTION 3 - ACHIEVING SOFTWARE ENGINEERING GOALS

SUMMARY OF MAIN POINTS COVERED:

- SUMMARY OF THE MAJOR METHODS AND MANAGEMENT TECHNIQUES WE CAN USE TO ACHIEVE ONE'S SOFTWARE ENGINEERING GOALS

MAIN MESSAGES:

- A COMBINATION OF METHODS AND TECHNIQUES ARE NEEDED TO ACHIEVE THESE GOALS
- TECHNICAL AND MANAGEMENT METHODS MUST BE USED IN COMBINATION TO ACHIEVE THESE GOALS

SUBTOPICS:

- SOFTWARE LIFE CYCLE
- INTRODUCTION TO THE METHODS
- METHODS FOR EACH PHASE OF THE LIFE CYCLE
- SOFTWARE MANAGEMENT METHODS AND TECHNIQUES

SPECIAL CONSIDERATIONS:

- KEEP INTRODUCTION SHORT

INSTRUCTOR NOTES

EMPHASIZE THAT THIS IS THE LIFE CYCLE MODEL USED IN THE MILITARY STANDARDS.

SECTION 3A - SOFTWARE LIFE CYCLE

SUMMARY OF MAIN POINTS COVERED:

- REVIEW THE TRADITIONAL SOFTWARE LIFE CYCLE MODEL AND ITS LIMITATIONS

MAIN MESSAGES:

- SOFTWARE DOES HAVE A LIFE THAT IS GREATER THAN JUST CODING
- NO UNIFORM VIEW OF THE LIFE CYCLE CURRENTLY EXISTS

SUBTOPICS:

- THE LIFE OF SOFTWARE
- A LIFE CYCLE MODEL (THE TRADITIONAL VIEW)
- SOFTWARE MAINTENANCE ACTIVITIES
- SHORTCOMINGS OF THE TRADITIONAL VIEW

SPECIAL CONSIDERATIONS:

- EMPHASIZE THAT THE REMAINDER OF SECTION 3 WILL FOLLOW THIS LIFE CYCLE MODEL

INSTRUCTOR NOTES

- EXPLAIN THE FORMAT USED IN THE PRESENTATION OF THE METHODS
  - INTRODUCTORY GRAPHICS
  - UNDERLYING CONCEPTS
  - HOW THEY SATISFY THE GOALS AND PRINCIPLES

SECTION 3B - INTRODUCTION TO METHODS AND TOOLS

SUMMARY OF MAIN POINTS COVERED:

- SUMMARIZES THE MAIN CHARACTERISTICS OF METHODOLOGIES THAT ARE USED IN ACHIEVING THE SOFTWARE ENGINEERING GOALS

MAIN MESSAGES:

- GOOD METHODS ARE IMPORTANT IN ACHIEVING THE SOFTWARE ENGINEERING GOALS
- THERE ARE REQUIREMENTS FOR AN IDEAL METHODOLOGY THAT YOU CAN MEASURE OTHER METHODOLOGIES AGAINST

SUBTOPICS:

- OUR SCOPE OF CONTROL
- ATTRIBUTES OF METHODOLOGIES
- WHY LEARN METHODOLOGIES
- REQUIREMENTS FOR AN IDEAL METHODOLOGY

SPECIAL CONSIDERATIONS:

- THESE TOPICS ARE ABSTRACT SO ADD PERSONAL EXPERIENCES TO MAKE THEM REAL

INSTRUCTOR NOTES

SECTION 3C - ANALYSIS OVERVIEW

SUMMARY OF MAIN POINTS COVERED:

- CHARACTERIZES THE ANALYSIS PHASE OF THE LIFE CYCLE

MAIN MESSAGES:

- WHAT ANALYSIS IS AND WHERE IT FITS INTO THE LIFE CYCLE

SUBTOPICS:

- DEFINITIONS
- ROLES OF INDIVIDUALS
- CONSEQUENCES OF WRONG REQUIREMENTS

SPECIAL CONSIDERATIONS:

- EMPHASIZE THE CONSEQUENCES OF WRONG REQUIREMENTS

INSTRUCTOR NOTES

RELATE TO THE INSTRUCTORS IN TRAINING THAT THEY SHOULD NOT ADVOCATE ANY PARTICULAR METHODOLOGY, THAT ALL OF THEM HAVE A PLACE IN THE DEVELOPMENT OF SOFTWARE.

SECTION 3D - ANALYSIS METHODS

SUMMARY OF MAIN POINTS COVERED:

- SUMMARY OF SOME POPULAR ANALYSIS METHODOLOGIES

MAIN MESSAGES:

- EACH METHOD ADDRESSES THE GOALS AND PRINCIPLES OF SOFTWARE ENGINEERING FROM A DIFFERENT PERSPECTIVE

SUBTOPICS:

- SADT
- SREM
- PSL/PSA
- SSA
- SCRP

SPECIAL CONSIDERATIONS:

- THE OBJECTIVE OF THIS SUBSECTION IS TO PROVIDE THE STUDENT WITH ENOUGH INFORMATION TO IDENTIFY THE METHODS AND TO UNDERSTAND HOW THEY ADDRESS THE GOALS OF SECTION 2
- THE MODE OF PRESENTATION FOR EACH METHOD IS
  - SLIDE SHOWING GRAPHICS OF TEMPLATES USED IN THE METHOD
  - SLIDE THAT SUMMARIZES THE CHARACTERISTICS OF METHOD
  - SLIDE IDENTIFYING THE GOALS AND PRINCIPLES THE METHOD SUPPORTS

INSTRUCTOR NOTES

THIS DESIGN OVERVIEW COVERS BOTH ARCHITECTURAL AND DETAILED DESIGN.

SECTION 3E - DESIGN OVERVIEW

SUMMARY OF MAIN POINTS COVERED:

- CHARACTERIZES THE DESIGN PHASE(S) OF THE LIFE CYCLE

MAIN MESSAGES:

- WHAT DESIGN IS AND WHERE IT FITS INTO THE LIFE CYCLE

SUBTOPICS:

- DEFINITIONS
- ROLE OF THE DESIGNER

SPECIAL CONSIDERATIONS:

- GO OVER THIS QUICKLY, GET STARTED ON THE METHODS, THAT IS WHERE THE REAL MEAT OF THE PRESENTATION IS

INSTRUCTOR NOTES

SECTION 3F - ARCHITECTURAL DESIGN METHODS

SUMMARY OF MAIN POINTS COVERED:

- SUMMARY OF SOME POPULAR ARCHITECTURAL DESIGN METHODOLOGIES

MAIN MESSAGES:

- EACH METHOD ADDRESSES THE GOALS AND PRINCIPLES OF SOFTWARE ENGINEERING FROM A DIFFERENT PERSPECTIVE

SUBTOPICS:

- SCRP
- OBJECT ORIENTED DESIGN
- STRUCTURED DESIGN
- JACKSON AND WARNIER ORR
- HIGHER ORDER SOFTWARE

SPECIAL CONSIDERATIONS:

- THE OBJECTIVE OF THIS SUBSECTION IS TO PROVIDE THE STUDENT WITH ENOUGH INFORMATION TO IDENTIFY THE METHODS AND TO UNDERSTAND HOW THEY ADDRESS THE GOALS OF SECTION 2
- THE MODE OF PRESENTATION FOR EACH METHOD IS
  - SLIDE SHOWING GRAPHICS OR TEMPLATES USED IN THE METHOD
  - SLIDE THAT SUMMARIZES THE CHARACTERISTICS OF METHOD
  - SLIDE IDENTIFYING THE GOALS AND PRINCIPLES THE METHOD SUPPORTS

INSTRUCTOR NOTES

SECTION 3G - DETAILED DESIGN METHODS

SUMMARY OF MAIN POINTS COVERED:

- SUMMARY OF SOME POPULAR DETAILED DESIGN METHODOLOGIES

MAIN MESSAGES:

- EACH METHOD ADDRESSES THE GOALS AND PRINCIPLES OF SOFTWARE ENGINEERING FROM A DIFFERENT PERSPECTIVE

SUBTOPICS:

- PROGRAM DESIGN LANGUAGES
- HIPO
- NSSF

SPECIAL CONSIDERATIONS:

- THE OBJECTIVE OF THIS SUBSECTION IS TO PROVIDE THE STUDENT WITH ENOUGH INFORMATION TO IDENTIFY THE METHODS AND TO UNDERSTAND HOW THEY ADDRESS THE GOALS OF SECTION 2
- THE MODE OF PRESENTATION FOR EACH METHOD IS
  - SLIDE SHOWING GRAPHICS OR TEMPLATES USED IN THE METHOD
  - SLIDE THAT SUMMARIZES THE CHARACTERISTICS OF METHOD
  - SLIDE IDENTIFYING THE GOALS AND PRINCIPLES THE METHOD SUPPORTS

INSTRUCTOR NOTES

THE POINT ABOUT IMPLEMENTATION BEING MORE THAN CODING IS VERY IMPORTANT SO MAKE IT CLEAR.

SECTION 3H - IMPLEMENTATION OVERVIEW

SUMMARY OF MAIN POINTS COVERED:

- CHARACTERIZES THE IMPLEMENTATION PHASE OF THE LIFE CYCLE

MAIN MESSAGES:

- WHAT IMPLEMENTATION IS AND WHERE IT FITS INTO THE LIFE CYCLE

SUBTOPICS:

- SCOPE OF IMPLEMENTATION PHASE
- IMPLEMENTATION ISSUES

SPECIAL CONSIDERATIONS:

- NONE

INSTRUCTOR NOTES

SECTION 3I - IMPLEMENTATION METHODS

SUMMARY OF MAIN POINTS COVERED:

- SUMMARY OF STRUCTURED DESIGN TECHNIQUES AND TESTING STRATEGIES

MAIN MESSAGES:

- EACH OF THE TECHNIQUES ADDRESS THE GOALS AND PRINCIPLES OF SOFTWARE ENGINEERING FROM A DIFFERENT PERSPECTIVE

SUBTOPICS:

- MOTIVATION/PURPOSE OF STRUCTURED DESIGN
- STRUCTURED DESIGN CONCEPTS AND GUIDELINES
- TESTING AS AN ERROR REMOVAL TECHNIQUE
- UNIT TESTING
- INTEGRATION STRATEGIES

SPECIAL CONSIDERATIONS:

- EMPHASIZE THAT IMPLEMENTATION IS MORE THAN CODING

INSTRUCTOR NOTES

THIS SECTION HAS THE MOST DETAILED COVERAGE OF A TOPIC AREA OF M101 SO BE PREPARED TO HAVE TO DO MORE PREPARATION FOR IT THAN THE OTHER SECTIONS.

SECTION 3J - SOFTWARE MANAGEMENT

SUMMARY OF MAIN POINTS COVERED:

- SUMMARY OF THE MAJOR ISSUES AND TECHNIQUES ASSOCIATED WITH THE MANAGEMENT OF SOFTWARE DEVELOPMENT

MAIN MESSAGES:

- SEVERAL TECHNIQUES ARE AVAILABLE TO SUPPORT THE MANAGEMENT OF THE PRODUCTS AND THE PROCESS OF SOFTWARE DEVELOPMENT

SUBTOPICS:

- SOFTWARE PLANNING AND TRACKING
- PLANNING TECHNIQUES
- SOFTWARE COST ESTIMATION
- SOFTWARE QUALITY MANAGEMENT
- SOFTWARE CONFIGURATION MANAGEMENT

SPECIAL CONSIDERATIONS:

- THE MANNER OF PRESENTATION OF THIS MATERIAL WILL VARY DEPENDING ON THE CLASS, SO BE PREPARED TO ACCELERATE IF THE CLASS IS MADE UP OF EXPERIENCED SOFTWARE MANAGERS

INSTRUCTOR NOTES

SECTION 4 - SOFTWARE ENGINEERING AND Ada

SUMMARY OF MAIN POINTS COVERED:

- HIGHLIGHTS THE RELATIONSHIP OF Ada AND SOFTWARE ENGINEERING

MAIN MESSAGES:

- Ada "THE LANGUAGE" IS BUILT TO ADDRESS THE SOFTWARE ENGINEERING GOALS AND BUILT ON THE FOUNDATION OF SOFTWARE ENGINEERING PRINCIPLES

SUBTOPICS:

- ANALYSIS AND Ada
- DESIGN AND Ada
- STRUCTURED DESIGN AND Ada
- HOW Ada SUPPORTS THE GOALS AND PRINCIPLES

SPECIAL CONSIDERATIONS:

- KEEP THIS DISCUSSION GENERAL. AVOID TALKING ABOUT Ada FEATURES DIRECTLY OR YOU WILL LOSE THE CLASS.

INSTRUCTOR NOTES

- ALLOW 80 MINUTES FOR THIS SECTION

SECTION 4

L201

Ada FOR SOFTWARE MANAGERS

INSTRUCTOR NOTES

GENERAL COMMENTS

- IN-CLASS EXERCISES ARE SCATTERED THROUGHOUT THE MODULE
- THERE ARE NO LAB EXERCISES
- SPECIAL CONSIDERATIONS ARE NOTED WHERE APPLICABLE, BASED ON PRIOR EXPERIENCE TEACHING THE MATERIAL
- EACH SECTION IS DISCUSSED, ITS MAIN POINTS AND ITS RELATION TO THE COURSE OVERALL

INSTRUCTOR NOTES

- GIVE THE INSTRUCTORS IN TRAINING AN OVERVIEW OF WHAT IS COVERED IN L201 AND WHAT ITS OBJECTIVES ARE.

- TARGET TEACHING TIME IS IN PARENTHESES
    - INCLUDED TO GIVE INSTRUCTORS IN TRAINING AN IDEA OF HOW MUCH TIME IS DEVOTED TO VARIOUS TOPICS
    - EMPHASIZE THAT TIME IS TARGET. MAY VARY DEPENDING ON CLASS NEEDS/BACKGROUND

- MENTION THAT DATA TYPES RECEIVE SO MUCH ATTENTION BECAUSE
    - TYPING IS NEW TO MANY STUDENTS
    - FUNDAMENTAL CONCEPT IN Ada

OVERVIEW

SECTION 1 - INTRODUCTION (:45)

REVIEWS AN Ada SYSTEM AND ITS IMPLEMENTATION

USE OF Ada FEATURES

SECTION 2 - LEXICAL RULES (:30)
SECTION 3 - DATA TYPES (3:00)
SECTION 4 - STATEMENTS (1:15)
SECTION 5 - PACKAGES/PRIVATE TYPES (1:15)
SECTION 6 - SUBPROGRAMS (1:15)
SECTION 7 - TASKS (:45)
SECTION 8 - GENERICS (:60)
SECTION 9 - I/O (:45)
SECTION 10 - EXCEPTIONS (1:15)
SECTION 11 - STUBBING (:60)
SECTION 12 - VISIBILITY AND SCOPE (:30)
SECTION 13 - OVERLOADING (:30)
SECTION 14 - PRAGMAS (:15)
SECTION 15 - LOW-LEVEL FEATURES (:45)
SECTION 16 - SUMMARY OF USES OF Ada FEATURES (:10)

SECTION 17 - INTRODUCTION TO Ada DESIGN/CODE ASSESSMENT (2:30)
STUDENTS DESIGN AND IMPLEMENT AN Ada SYSTEM, AND DISCUSS EXPERIENCES

SECTION 18 - CHARACTERISTICS OF GOOD Ada DESIGNS (:30)
FORMALIZES IDEAS AND CONCEPTS OF PREVIOUS SECTION

SECTION 19 - Ada IN PERSPECTIVE : REUSABILITY AND PORTABILITY (:30)
HIGH LEVEL DISCUSSION OF REUSABILITY AND PORTABILITY ISSUES

INSTRUCTOR NOTES

- MANAGERS MUST UNDERSTAND THE PURPOSE OF THIS MODULE
  - PREPARES THEM TO EVALUATE DESIGNS
  - RESOLVE USAGE DISPUTES

MANAGERS NEEDING/WANTING ALL THE DETAILS SHOULD BE TAKING L202, L305 AND L401

SECTION 1 - INTRODUCTION

SUMMARY OF MAIN POINTS COVERED:

- REVIEW OF Ada LANGUAGE OVERVIEW FROM L101
- REVIEW OF Ada COMPILATION SYSTEM FROM L101

MAIN MESSAGES:

- POWERFUL Ada FEATURES MUST BE USED SENSIBLY
- MANAGERS MAY BE CALLED UPON TO DECIDE WHETHER OR NOT CERTAIN FEATURES ARE USED AND, IF SO, HOW THEY ARE TO BE USED

SUBTOPICS:

- STRUCTURE OF A SIMPLE Ada SYSTEM
- COMPILATION SYSTEM

SPECIAL CONSIDERATIONS:

- MAKE SURE THE STUDENTS UNDERSTAND THEY ARE NOT GETTING ALL THE DETAILS OF THE LANGUAGE
  - PRIMARY CONCEPTS
  - DETAILS ARE FOR PROGRAMMERS
  - WILL BE ABLE TO SETTLE STYLE AND USAGE DISPUTES
  - WILL BE ABLE TO EVALUATE Ada-BASED DESIGNS

INSTRUCTOR NOTES

- SECTIONS 2-15 PRESENT THE Ada FEATURES

- THE SUMMARY OF MAIN POINTS COVERED, MAIN MESSAGES, SUBTOPICS, AND SPECIAL CONSIDERATIONS LISTED ON THIS SLIDE APPLY TO EACH OF THE ABOVE SECTIONS. THEY WILL NOT BE REPEATED FOR EACH SECTION. SUBSEQUENT SLIDES WILL EXPAND THESE HEADINGS.

USE OF Ada FEATURES

SUMMARY OF MAIN POINTS COVERED:

- PROVIDES CONCEPTUAL KNOWLEDGE OF THE FEATURES OF THE Ada LANGUAGE

MAIN MESSAGES:

- Ada FEATURES NEED TO BE USED WITH PURPOSEFUL INTENT AND KNOWLEDGE

SUBTOPICS:

- FOR EACH FEATURE
  - INTENDED USES OF THE FEATURE
  - LANGUAGE RULES
  - EXAMPLES OF THE INTENDED USE
  - PITFALLS OR MISUSES CONNECTED WITH THE FEATURES

SPECIAL CONSIDERATIONS:

- IT IS IMPORTANT FOR TECHNICAL MANAGERS TO KNOW HOW Ada FEATURES SHOULD OR SHOULD NOT BE USED

INSTRUCTOR NOTES

- INSTRUCTORS IN TRAINING SHOULD READ THIS CASE STUDY AND SHOULD RECOMMEND IT TO MANAGERS.
    - SUGGEST THAT IDEAS PRESENTED THERE WILL GREATLY INCREASE READABILITY AND MAINTAINABILITY OF CODE
    - MANAGER'S PROJECT PROGRAMMERS SHOULD ALSO READ THIS

SECTION 2 - LEXICAL RULES

SUMMARY OF MAIN POINTS COVERED:

- THE MOST COMMON USES AND PITFALLS OF LEXICAL ELEMENTS

MAIN MESSAGES:

- LEXICAL ELEMENTS ARE THE BUILDING BLOCKS OF Ada PROGRAMS
- COMMON PITFALLS ARE CRYPTIC CHOICE OF ENTITY NAMES, COMMENTS, OR LACK OF FORMATTING TO REFLECT LOGIC
- GOOD CHOICE OF IDENTIFIER NAMES, PROPER FORMATTING, AND APPROPRIATE COMMENTS INCREASE READABILITY AND MAINTAINABILITY

SUBTOPICS:

- IDENTIFIERS
- NUMERIC LITERALS
- CHARACTER AND STRING LITERALS
- DELIMITERS
- COMMENTS

SPECIAL CONSIDERATIONS:

- CASE STUDY "GUIDELINES FOR THE SELECTION OF IDENTIFIERS" IN Ada CASE STUDIES II REPORT CONTRACT NUMBER DAAB07-83-C-K514, CECOM, DECEMBER 1983. PROVIDES EXCELLENT DISCUSSION OF NAMING CONVENTIONS.

INSTRUCTOR NOTES

- FOR MANY MANAGERS, THIS MAY BE THE MOST DIFFICULT SECTION.
  - IF ONLY EXPERIENCE IS ASSEMBLER OR FORTRAN, MAY HAVE TROUBLE UNDERSTANDING WHY Ada'S TYPING SYSTEM EXISTS
  - MANAGERS WILL NOT HAVE Ada CODING EXERCISES, SO THEY WILL NOT GET FIRST HAND EXPERIENCE
  - MUST STRESS READABILITY, AND COMPILE-TIME ERROR DETECTION

SECTION 3 - DATA TYPES

SUMMARY OF MAIN POINTS COVERED:

- THE MOST COMMON USES AND PITFALLS OF PREDEFINED AND USER-DEFINED TYPES

MAIN MESSAGES:

- PREDEFINED TYPES ARE THE BASIC DATA TYPES SIMILAR TO THOSE OF OTHER LANGUAGES
- IN GENERAL, USER-DEFINED TYPES ALLOW US TO EXPRESS AN ALGORITHM IN A MORE NATURAL WAY
- COMMON PITFALLS INCLUDE
  - OVERUSE OF PREDEFINED TYPES INSTEAD OF USER-DEFINED TYPES
  - NOT BUILDING THE APPROPRIATE USER DEFINED TYPE

SUBTOPICS:

- OBJECTS
- INTEGER TYPE
- FLOATING POINT AND FIXED POINT
- NUMERIC CONVERSIONS
- ATTRIBUTES
- SUBTYPES
- ENUMERATION TYPES
- ARRAY TYPES
- RECORDS/DISCRIMINANTS
- AGGREGATES
- ACCESS TYPES
- DERIVED TYPES
- STRONG TYPING

INSTRUCTOR NOTES

- SUB-BULLET 4
  - EXAMPLE NAVIGATION DATA
    - DO NOT USE TWO PARALLEL ARRAYS, ONE FOR LATITUDE AND ONE FOR LONGITUDE
    - USE AN ARRAY OF POSITION RECORDS

SPECIAL CONSIDERATIONS:

- POINTS TO EMPHASIZE ARE:
  - USING ATTRIBUTES DECREASES IMPLEMENTATION DEPENDENCIES AND INCREASES READABILITY AND MAINTAINABILITY
  - ENUMERATION TYPES CAN OFTEN BE USED INSTEAD OF AN INTEGER TYPE
    - ESPECIALLY EMPHASIZE AS INDEX TYPE
  - RECORDS WITH DISCRIMINANTS ALLOW OBJECTS WITH DIFFERENT SIZES/STRUCTURES
  - RECORDS ALLOW LOGICALLY RELATED OBJECTS TO BE GROUPED TOGETHER
    - EMPHASIZE MUST *UNLEARN* "FORTRAN" WAY OF RELATING LOGICALLY RELATED DATA
  - EMPHASIZE STRONG TYPING
    - BIGGEST HURDLE FOR PROGRAMMERS INTRODUCED TO IT FOR THE FIRST TIME
    - PAYOFF IS WORTH IT
    - FORCES PROGRAMMERS TO THINK OF LOGICAL VIEW OF OBJECTS
    - COMPILER DETECTS VIOLATIONS OF THESE VIEWS

INSTRUCTOR NOTES

- IF THE MANAGERS HAVE A GOOD DEAL OF HOL EXPERIENCE, THEN DO NOT SPEND MUCH TIME ON THE MECHANICS OF THESE STATEMENTS.

SECTION 4 - STATEMENTS

SUMMARY OF MAIN POINTS COVERED:

- THE COMMON USES AND MISUSES OF STATEMENTS

MAIN MESSAGES:

- STATEMENTS CONTROL THE FLOW OF LOGIC
- COMMON PITFALLS INCLUDE NOT USING ATTRIBUTES TO CONTROL A LOOP, USING THE WRONG LOOP, USING goto RATHER THAN exit
- APPROPRIATE CHOICE OF STATEMENTS IMPROVES READABILITY AND MAINTAINABILITY OF CODE

SUBTOPICS:

- ASSIGNMENT STATEMENT
- NULL STATEMENT
- IF STATEMENT
- CASE STATEMENT
- SIMPLE LOOP STATEMENTS
- FOR LOOP STATEMENT
- WHILE LOOP STATEMENT
- EXIT STATEMENT
- GOTO STATEMENT
- RETURN STATEMENT
- PROCEDURE CALL STATEMENT

INSTRUCTOR NOTES

- SUB-BULLET 1
  - A WHILE LOOP OR SIMPLE LOOP CAN BE USED IN PLACE OF A FOR LOOP, BUT INTENT IS LESS CLEAR

- SUB-BULLET 4
  - THE FOLLOWING EXAMPLE WITH THE exit STATEMENT IS INCLUDED JUST AS AN EXAMPLE FOR THE INSTRUCTORS. IT SHOWS WHY A LOOP MIGHT BE LEFT IN THE MIDDLE. THIS ENSURES THAT THEY HAVE AN EXAMPLE TO FALL BACK ON WHEN TEACHING THE MODULE.

```
Loop_Identification
   loop
      -- code for each pass through the loop
      if [condition] then
         exit Loop_Identification;
      -- code to execute if [condition] does not hold
   end loop Loop_Identification;
```

  - THIS IS OFTEN CALLED A LOOP AND A HALF

- LAST SUB-BULLET
  - MAKE SURE THAT INSTRUCTORS IN TRAINING REALIZE THAT THEY SHOULD GO EASY ON THE goto STATEMENT. IT IS NOT THE "ROOT OF ALL EVIL"! Ada FEATURES PROVIDE MORE ATTRACTIVE WAYS TO ACCOMPLISH "goto's". HOWEVER, THERE ARE TIMES WHERE NOT USING A goto CAN RESULT IN CONVOLUTED LOGIC. DO NOT BE FANATICAL ABOUT NOT USING goto's. JUST BE CAREFUL WHEN A PROGRAMMER SAYS "I NEED THE goto." MAKE THE PROGRAMMER SHOW YOU WHY IT IS NEEDED!

SPECIAL CONSIDERATIONS:

- EMPHASIZE THAT APPROPRIATE CHOICE OF STATEMENTS CAN MAKE CODE MUCH EASIER TO READ AND MAINTAIN
  - FOR LOOP WHEN ITERATION IS NEEDED
    - USE ATTRIBUTES TO CONTROL THE LOOP
    - DON'T BE AFRAID TO USE WITH ENUMERATION TYPES
  - elsif WHEN SELECTING ALTERNATIVES RATHER THAN NESTED IFS
  - CASE STATEMENT WHEN SELECTING FROM MUTUALLY EXCLUSIVE DISCRETE ALTERNATIVES
  - USING EXIT TO LEAVE A LOOP RATHER THAN A GOTO
    - SOMETIMES NEED TO EXIT LOOP PREMATURELY
  - NOT USING A goto WHEN ONE IS CALLED FOR
    - GENERALLY DO NOT NEED goto BECAUSE OF Ada CONDITIONAL AND LOOP STATEMENTS , EXIT AND RETURN STATEMENTS, SUBPROGRAMS AND EXCEPTION HANDLING
    - SHOULD NOT EXCLUDE IF RESULT IS CONVOLUTED LOGIC

INSTRUCTOR NOTES

SECTION 5 - PACKAGES AND PRIVATE TYPES

SUMMARY OF MAIN POINTS COVERED:

- THE MOST COMMON USES AND PITFALLS OF PACKAGES AND PRIVATE TYPES

MAIN MESSAGES:

- PACKAGES ARE AN IMPORTANT DESIGN TOOL
- PACKAGES PROVIDE A WAY OF GROUPING RELATED ENTITIES
- PACKAGES DISTINGUISH INTERFACE FROM IMPLEMENTATION
- PACKAGES PROVIDE FOR REUSABLE SOFTWARE COMPONENTS
- PRIVATE TYPES SUPPORT DATA ABSTRACTION
- PRIVATE TYPES PREVENT USE OF TYPE FROM USING REPRESENTATION
- COMMON PITFALLS INCLUDE NOT USING PACKAGES, HAVING UNRELATED ENTITIES IN A PACKAGE, AND USING PACKAGE AS FORTRAN COMMON

SUBTOPICS:

- ABSTRACT DATA TYPES
- INFORMATION HIDING
- LIMITED PRIVATE TYPES

INSTRUCTOR NOTES

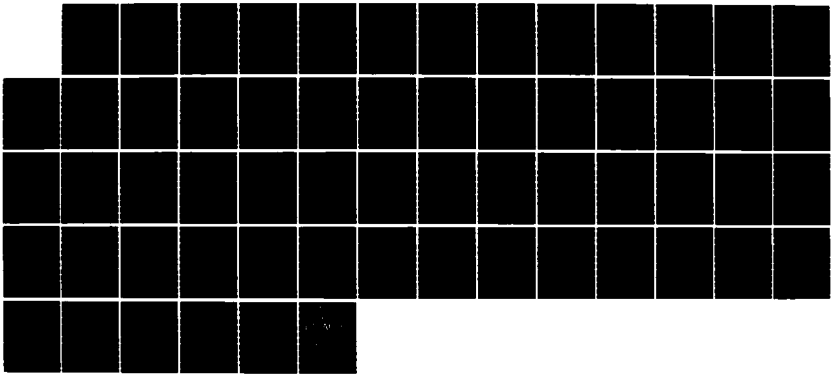
- REUSABILITY IS DISCUSSED MORE FULLY IN SECTION 19, BUT IT IS SUCH AN IMPORTANT CONCEPT THAT INSTRUCTORS IN TRAINING SHOULD FEEL FREE TO DISCUSS IT A LITTLE HERE.

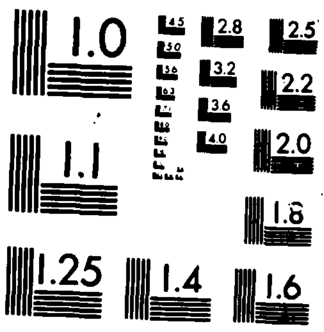
- MANAGERS MUST UNDERSTAND THE ADVANTAGES OF HAVING SPECIFICATION PART AND IMPLEMENTATION PART.

- LAST SUB-BULLET - PRIVATE TYPES
  - DIFFERENT IMPLEMENTATIONS OF THE SAME ABSTRACTION CAN BE PROVIDED
  - VERSIONS TO SUPPORT TESTING
  - PRODUCTION VERSIONS

AD-A165 347 ADA (TRADEMARK) TRAINING CURRICULUM INSTRUCTOR'S COURSE 2/2
S500 MANAGEMENT MODULES(U) SOFTECH INC WALTHAM MA 1986
DAAB07-85-C-K506

UNCLASSIFIED F/G 5/9 NL




1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

SPECIAL CONSIDERATIONS:

- POINTS TO EMPHASIZE ARE
  - PACKAGES PROVIDE FOR REUSABLE SOFTWARE COMPONENTS
    - ENCOURAGE GENERAL PURPOSE PACKAGES
      - SHARE ACROSS PROJECTS
      - COST EFFECTIVE
      - ESTABLISH GENERAL PURPOSE LIBRARIES OF HIGH QUALITY PACKAGES
  - PACKAGE HAS
    - SPECIFICATION PART
      - THIS IS ALL A PROGRAMMER NEEDS TO KNOW TO USE SERVICES PROVIDED BY PACKAGE
      - RELIANCE ON DETAILS OF MANIPULATION SEPARATED FROM USE
    - IMPLEMENTATION PART
      - IMPLEMENTOR VIEWS SPECIFICATION AS CONTRACT BETWEEN USERS AND SELF
  - PRIVATE TYPES PROVIDE PURELY ABSTRACT VIEW OF PACKAGE SERVICES
    - IMPLEMENTATION DETAILS HIDDEN AS BEFORE
    - ALSO ALLOWS REPRESENTATION TO BE HIDDEN
    - USERS CANNOT "ACCIDENTALLY" USE DETAILS OF TYPES PROVIDED BY PACKAGE

INSTRUCTOR NOTES

- AGAIN, SUGGEST MANAGERS READ THE FIRST CASE STUDY, "GUIDELINES FOR THE SELECTION OF IDENTIFIERS".

SECTION 6 - SUBPROGRAMS

SUMMARY OF MAIN POINTS COVERED:

- THE MOST COMMON USES AND PITFALLS OF SUBPROGRAMS

MAIN MESSAGES:

- SUBPROGRAMS ARE ALGORITHMIC ABSTRACTIONS
- SUBPROGRAMS SHOULD NOT BE USED AS TO BE DEFINED LATER

SUBTOPICS:

- PROCEDURES
- FUNCTIONS
- MAIN PROGRAM
- PARAMETER MODES
- NAMED PARAMETERS
- SUBPROGRAMS AS OPERATOR ON A TYPE

SPECIAL CONSIDERATIONS:

- NAMED SUBPROGRAM PARAMETERS SHOULD BE EMPHASIZED AS INCREASING READABILITY AND MAINTAINABILITY
- POINT OUT THAT SUBPROGRAM SPECIFICATIONS INDICATE THE SERVICE TO BE PERFORMED, <u>NOT</u> HOW THE SERVICE IS PERFORMED
- EMPHASIZE THE ROLE OF SUBPROGRAMS ON A TYPE
- CASE STUDY "GUIDELINES FOR THE SELECTION OF IDENTIFIERS" IN Ada CASE STUDIES II REPORT SUGGESTS WAY OF NAMING SUBPROGRAMS AND THEIR PARAMETERS TO PROVIDE FOR "ENGLISH LIKE" READING OF SUBPROGRAM CALLS
- EMPHASIZE THE USE OF NAMED PARAMETERS OVER POSITIONAL ONES

INSTRUCTOR NOTES

- INSTRUCTORS IN TRAINING SHOULD REALIZE THAT Ada TASKING WILL PROBABLY BE OF INTEREST TO MOST MANAGERS. THEY SHOULD EXPECT LOTS OF QUESTIONS ABOUT TASKING, MOST OF WHICH MUST BE DEFERRED TO L303.

- THE PROBLEM OF DEADLOCK IS A SERIOUS ONE. IT IS DISCUSSED IN L401.

SECTION 7 - TASKS

SUMMARY OF MAIN POINTS COVERED:

- THE MOST COMMON USES AND PITFALLS OF TASKS

MAIN MESSAGES:

- TASKS ALLOW THE EXPRESSION OF CONCURRENT ACTIONS WITHIN AN Ada PROGRAM
- TASKS CAN BE USED FOR CONTROLLING RESOURCES
- TASKS CAN BE USED AS INTERRUPT HANDLERS
- DEADLOCK CAN OCCUR IF CARE IS NOT TAKEN

SUBTOPICS:

- TASK TYPES
- RENDEZVOUS
- SELECTIVE WAIT STATEMENT
- INTERRUPT ENTRIES

SPECIAL CONSIDERATIONS:

- THIS SECTION IS NECESSARILY A VERY HIGH LEVEL OVERVIEW
  - LET MANAGERS KNOW THERE ARE SUCH THINGS AS TASKS
  - LET MANAGERS KNOW TASKS CAN BE USED TO DO INTERESTING THINGS
  - LET MANAGERS KNOW TASK INTERACTION CAN BE CONTROLLED BY USER
- THERE IS A STRONG TEMPTATION TO GO INTO DETAILS ABOUT RENDEZVOUS, SELECTIVE WAITS, INTERRUPT ENTRIES, ETC.
  - AVOID DOING SO
  - DIRECT INTERESTED MANAGERS TO L303 - THAT'S WHAT IT'S THERE FOR!
  - ENCOURAGE MANAGERS TO TAKE L303
- DEADLOCK IS DISCUSSED IN L401

INSTRUCTOR NOTES

SECTION 8 - GENERICS

SUMMARY OF MAIN POINTS COVERED:

- THE MOST COMMON USES AND PITFALLS OF GENERICS

MAIN MESSAGES:

- GENERIC UNITS ALLOW FOR PARAMETERIZED TEMPLATES
- GENERIC UNITS ENHANCE READABILITY
- GENERIC UNITS PROVIDE EFFECTIVE REUSE OF COMMON CODE AND ENCOURAGE BUILDING OF REUSABLE SOFTWARE COMPONENT LIBRARIES
- COMMON PITFALL IS TO CODE NEW PROGRAM UNIT WHEN THERE ALREADY EXISTS A PROGRAM UNIT DIFFERING ONLY IN VARIABLE TYPES, SIZE OF STRUCTURES OR SUBPROGRAMS

SUBTOPICS:

- GENERIC FORMAL PARAMETERS
- GENERIC ACTUAL PARAMETERS
- INSTANTIATION

INSTRUCTOR NOTES

- AGAIN, THE TOPIC OF REUSABILITY SHOULD BE DISCUSSED.
- INSTRUCTORS IN TRAINING SHOULD ENCOURAGE MANAGERS TO FORM LIBRARIES OF REUSABLE GENERIC PROGRAM UNITS.
- GENERIC INSTANTIATION EFFICIENCY SHOULD NOT BE MENTIONED BY INSTRUCTORS IN TRAINING. HOWEVER, SOME BAD PRESS ABOUT THIS TOPIC HAS APPEARED. THE MATERIAL PRESENTED HERE IS PROVIDED FOR INSTRUCTORS IN TRAINING TO SET THE RECORD STRAIGHT IF THEY CAN'T AVOID THE TOPIC.

SPECIAL CONSIDERATIONS:

- EMPHASIZE REUSABILITY ASPECTS OF GENERICS
  - QUEUES, STACKS, SEARCH AND SORT ALGORITHMS ETC., APPEAR MANY TIMES IN PROGRAMS
  - MAY DIFFER IN SIZE (QUEUE SIZE, STACK SIZE, TABLE SIZE), TYPE (QUEUE, STACK OR TABLE ELEMENT TYPE) OR FUNCTION (ORDERING OR EQUALITY SUBPROGRAMS IN SEARCH OR SORT)
  - DETAILS "ESSENTIALLY THE SAME"
  - WASTE OF EFFORT TO REWRITE EACH TIME NEEDED
  - ENCOURAGE INTER-PROJECT REUSABLE LIBRARY
- GENERIC INSTANTIATION EFFICIENCY
  - DO NOT BRING THIS UP UNLESS ASKED ABOUT IT!
  - DEPENDS ON IMPLEMENTATION
    - SOME MAY USE MACRO APPROACH
    - OTHERS MAY USE RUNTIME-DESCRIPTORS
    - FOR SOME CASES, MAY NOT NEED TO DO EITHER
  - APPROACH USED MAY DEPEND ON
    - OPTIMIZATIONS PERFORMED BY COMPILER
    - SPACE OPTIMIZATIONS REQUESTED
    - TIME OPTIMIZATIONS REQUESTED
    - GENERIC FORMAL/ACTUAL PARAMETER TYPES
      - ALL DISCRETE
      - MIXED
    - MOST LIKELY NOT A PROBLEM
      - DO NOT BE CONCERNED WITH IT UNLESS PERFORMANCE SHOWS IT TO BE A PROBLEM
      - EASY TO FIX, IF NECESSARY

INSTRUCTOR NOTES

SECTION 9 - I/O

SUMMARY OF MAIN POINTS COVERED:

- THE MOST COMMON USES AND PITFALLS OF Ada I/O

MAIN MESSAGES:

- Ada I/O ALLOWS THE USER CONTROL OF APPLICATION I/O

SUBTOPICS:

- TEXT I/O
  - CHARACTER/STRINGS
  - GENERIC TEMPLATES:
    - INTEGER
    - ENUMERATION
    - FLOATING POINT
    - FIXED POINT
- SEQUENTIAL I/O
- DIRECT I/O

SPECIAL CONSIDERATIONS:

- EMPHASIZE HOW MUCH CONTROL USER HAS OVER I/O
- DO NOT GIVE DETAILED DISCUSSION OF I/O FEATURES

INSTRUCTOR NOTES

- THE CASE STUDY ON EXCEPTIONS SHOULD BE READ BY MANAGERS AND INSTRUCTORS IN TRAINING.
- EMPHASIZE THAT PROPER USE OF EXCEPTIONS CAN SIGNIFICANTLY INCREASE RELIABILITY.

SECTION 10 - EXCEPTIONS

SUMMARY OF MAIN POINTS COVERED:

- COMMON USES AND PITFALLS OF EXCEPTIONS

MAIN MESSAGES:

- USE OF EXCEPTIONS IS A HIGH-LEVEL DESIGN ISSUE
- EXCEPTIONS ARE A MECHANISM FOR FAULT-TOLERANT PROGRAMMING
- EXCEPTIONS CAN BE USED TO SUPPORT RELIABILITY
- EXCEPTIONS CAN BE USED TO RECOVER FROM HARDWARE MALFUNCTIONS

SUBTOPICS:

- HANDLING EXCEPTIONS
- PROPAGATING EXCEPTIONS
- PREDEFINED EXCEPTIONS
- USER DEFINE EXCEPTIONS
- RAISING EXCEPTIONS
- SUGGESTIONS FOR USING EXCEPTIONS

SPECIAL CONSIDERATIONS:

- EMPHASIZE USE OF EXCEPTIONS SHOULD BE PLANNED AT DESIGN STAGE
- EMPHASIZE BLOCK STATEMENTS AS A WAY TO LOCALIZE EFFECT OF EXCEPTIONS
- CARE SHOULD BE TAKEN NOT TO DEPEND ON A PARTICULAR PREDEFINED EXCEPTION BEING RAISED
  - A COMPUTATION MAY RAISE
    - Constraint_Error
    - Numeric_Error
- SEE CASE STUDY ON EXCEPTIONS

INSTRUCTOR NOTES

- EMPHASIZE THAT Ada STUBBING SUPPORTS TOP DOWN AND BOTTOM UP PROGRAMMING. MOREOVER, A COMBINATION OF THE TWO CAN BE SUPPORTED. THIS ALLOWS A MANAGER A GREAT DEAL OF LATITUDE IN MAKING WORK ASSIGNMENTS.

SECTION 11 - STUBBING

SUMMARY OF MAIN POINTS COVERED:

- THE MOST COMMON USES AND PITFALLS OF STUBBING

MAIN MESSAGES:

- STUBBING CAN DECREASE PHYSICAL COMPLEXITY
- STUBBING CAN BE USED TO INCREASE UNDERSTANDABILITY OF COMPLEX CODE AND DESIGN
- RECOMPILATION COSTS CAN BE DECREASED BY USING STUBBING
- OVERUSE CAN LEAD TO EXCESSIVE FRAGMENTATION AND LOSS OF UNDERSTANDABILITY

SUBTOPICS:

- TOP DOWN PROGRAMMING
- BOTTOM UP PROGRAMMING
- PROGRAM LIBRARY
- COMPILATION ORDER
- SEPARATE COMPILATION
- LIBRARY UNITS

SPECIAL CONSIDERATIONS:

- EMPHASIZE THAT Ada SUPPORTS BOTH TOP DOWN AND BOTTOM UP PROGRAMMING
- COMPILATION COSTS CAN BE SIGNIFICANTLY REDUCED BY STUBBING
- THE TERM "PROGRAM LIBRARY" UNFORTUNATELY CAUSES PEOPLE TO THINK OF CODE, AND THEREFORE THEY THINK THE PROGRAM LIBRARY CONTAINS CODE
  - EMPHASIZE <u>NO</u> CODE IN PROGRAM LIBRARY
  - THINK OF PROGRAM LIBRARY AS "INTERFACE LIBRARY"

INSTRUCTOR NOTES

- THE EXAMPLES BELOW GIVE THE INSTRUCTORS AN EXAMPLE TO FALL BACK ON. EMPHASIZE HOW UNCLEAR THE SECOND EXAMPLE IS. IT REQUIRES EFFORT TO DETERMINE WHETHER A IS A LOGICAL VARIABLE OR A FUNCTION CALL TO A. THE SAME PROBLEM WOULD OCCUR IF A WAS A VARIABLE GLOBAL TO B.

- INCREASED READABILITY : A SUBPROGRAM NAMED Get

```
-      Text_IO.Get (   )
-      Message_Package.Get (   )
```

- DECREASED READABILITY : FUNCTION A VS. VARIABLE A

```
function A ( ... ) return ... is
   procedure B ( ...) is
      A : ... ;
   begin
      X := A ( ...);
   end B;
begin
  .
  .
  .
end A;
```

SECTION 12 - VISIBILITY AND SCOPE

SUMMARY OF MAIN POINTS COVERED:

- THE MOST COMMON USES AND PITFALLS OF VISIBILITY AND SCOPE RULES

MAIN MESSAGES:

- VISIBILITY AND SCOPE RULES ALLOW FOR DELIBERATE SHARING OF NAMES WHILE ALLOWING THE SAME NAMES TO HAVE DIFFERENT MEANINGS IN OTHER PORTIONS OF A PROGRAM
- GENERALLY, THE use CLAUSE MAKES SOFTWARE LESS SELF-DOCUMENTING

SUBTOPICS:

- with CLAUSES
- use CLAUSES
- NESTED UNITS

SPECIAL CONSIDERATIONS:

- DISCOURAGE USE OF use CLAUSES
- USING THE SAME NAME IN DIFFERENT PORTIONS OF A PROGRAM CAN INCREASE READABILITY IF USED SENSIBLY BUT DECREASE IT OTHERWISE

INSTRUCTOR NOTES

SECTION 13 - OVERLOADING

SUMMARY OF MAIN POINTS COVERED:

- THE MOST COMMON USES AND PITFALLS OF OVERLOADING

MAIN MESSAGES:

- OVERLOADING ALLOWS MORE NATURAL NAMES OF ENTITIES
- AVOIDING OVERLOADING CAN RESULT IN PROGRAMS THAT ARE MORE DIFFICULT TO READ

SUBTOPICS:

- SUBPROGRAM OVERLOADING
- ENUMERATION LITERAL OVERLOADING
- QUALIFICATION
- OPERATOR OVERLOADING

SPECIAL CONSIDERATIONS:

- SOME MANAGERS MAY INITIALLY FEEL OVERLOADING MAY BE TOO CONFUSING
- EMPHASIZE THE INCREASED PROGRAM READABILITY AND MAINTAINABILITY OBTAINED WHEN NATURAL NAMES CAN BE USED THROUGH OVERLOADING
- EMPHASIZE THAT OPERATOR OVERLOADING SHOULD BE USED EXACTLY WHEN THERE IS A NATURAL MATHEMATICAL MEANING

INSTRUCTOR NOTES

SECTION 14 - PRAGMAS

SUMMARY OF MAIN POINTS COVERED:

- THE MOST COMMON USES AND PITFALLS OF PRAGMAS

MAIN MESSAGES:

- PRAGMAS PROVIDE "ADVICE" TO THE COMPILER
- SOME PRAGMAS SHOULD BE USED ONLY WITH CONSIDERABLE JUSTIFICATION

SUBTOPICS:

- PREDEFINED PRAGMAS
- IMPLEMENTATION DEFINED PRAGMAS
- PRAGMAS TO BE USED WITH CAUTION; INLINE, OPTIMIZE, PACK
- PRAGMAS TO BE USED WITH EXTREME CARE: SHARED, SUPPRESS

SPECIAL CONSIDERATIONS:

- POINTS TO EMPHASIZE ARE THE INLINE PRAGMA SHOULD BE USED ONLY WHEN PERFORMANCE TESTING SUGGESTS IT
  - PRAGMA INCREASE RECOMPILATION COSTS
  - USE ONLY AFTER SUBPROGRAM GOES INTO PRODUCTION USE
- Ada TASKS SHOULD NORMALLY COMMUNICATE THROUGH RENDEZVOUS
  - SHARED DATA SHOULD BE ACCESSED THROUGH MONITORS
  - PROBLEM OF SIMULTANEOUS UPDATE CAN OCCUR
- THE SUPPRESS PRAGMA INFORMS THE COMPILER THAT THE CONDITION CANNOT OCCUR
  - IT'S THE PROGRAMMER'S FAULT IF IT DOES
  - SHOULD NOT BE USED UNLESS IT CAN BE "DEMONSTRATED" THAT THE CONDITION DOES NOT OCCUR

INSTRUCTOR NOTES

- INSTRUCTORS IN TRAINING MUST EMPHASIZE THAT LOW-LEVEL FEATURES SHOULD BE USED IN AN ABSTRACT WAY.
  - ISOLATES LOW-LEVEL USE
  - INCREASES PORTABILITY

SECTION 15 - LOW-LEVEL FEATURES

SUMMARY OF MAIN POINTS COVERED:

- THE MOST COMMON USES AND PITFALLS OF LOW-LEVEL FEATURES

MAIN MESSAGES:

- LOW-LEVEL FEATURES CAN BE USED FOR INTERFACING WITH UNDERLYING HARDWARE, DEVICES OR EXISTING CODE
- A COMMON MISUSE OF LOW-LEVEL FEATURES IS THE LACK OF ENCAPSULATION OF THEIR USAGE

SUBTOPICS:

- ENUMERATION TYPE REPRESENTATION CLAUSES
- RECORD TYPE REPRESENTATION CLAUSES
- ADDRESS CLAUSES
- CODE PROCEDURES
- LOW-LEVEL I/O
- UNCHECKED CONVERSION

SPECIAL CONSIDERATIONS:

- EMPHASIZE THAT THE USE OF LOW-LEVEL FEATURES SHOULD BE ENCAPSULATED
  - PRESENT HIGH-LEVEL ABSTRACTION OF USE BUT LOW-LEVEL IMPLEMENTATION
  - EXAMPLE
    - HIGH-LEVEL PROCEDURE Operator_Message DISPLAYS MESSAGE ON OPERATOR CONSOLE
    - LOW-LEVEL IMPLEMENTATION MIGHT USE CODE PROCEDURES FOR LOADING REGISTERS AND ISSUING SVC INSTRUCTION OR IMPLEMENTATION MIGHT USE LOW-LEVEL I/O
- EMPHASIZE THAT Ada ALLOWS HIGH ORDER LANGUAGE TECHNIQUES TO BE APPLIED TO LOW-LEVEL PROGRAMMING

INSTRUCTOR NOTES

SECTION 16 - SUMMARY OF USES OF Ada FEATURES

SUMMARY OF MAIN POINTS COVERED:

- SUMMARY OF TYPICAL USES OF Ada FEATURES

MAIN MESSAGES:

- Ada PROVIDES FEATURES TO WRITE READABLE, MAINTAINABLE AND RELIABLE SOFTWARE

SPECIAL CONSIDERATIONS:

- THIS SHOULD BE USED AS A BRIEF REVIEW OF WHAT FEATURES ARE PROVIDED BY Ada AND WHAT THEY TYPICALLY ARE USED FOR

INSTRUCTOR NOTES

- THE EXAMPLE IS THAT OF A TELEPHONE DIRECTORY SYSTEM PROVIDING
  - LOOK-UP NUMBER FOR GIVEN NAME
  - ADD NEW NAMES AND NUMBERS
  - DELETE NAMES AND NUMBERS
- INSTRUCTORS IN TRAINING MUST BE VERY FAMILIAR WITH THIS SECTION. IT IS VERY DEMANDING ON THE INSTRUCTOR.
- THE MATERIAL IN THIS SECTION IS LOGICALLY PRESENTED IN THREE DISTINCT PARTS. THE NEXT THREE SLIDES PRESENT THESE THREE PARTS.

SECTION 17 - INTRODUCTION TO Ada DESIGN/CODE ASSESSMENT

SUMMARY OF MAIN POINTS COVERED:

- PROVIDES DIRECT EXPERIENCE IN DESIGN AND CODING CONSIDERATIONS IN Ada

MAIN MESSAGES:

- A SYSTEM DESIGNED IN Ada REQUIRES MORE PLANNING, BUT THE REWARDS - READABILITY, MAINTAINABILITY, RELIABILITY ARE WORTH IT
- EVEN THOUGH DESIGNING AN Ada SYSTEM TAKES MORE PLANNING TIME, IT CAN SIGNIFICANTLY DECREASE TESTING TIME

SUBTOPICS:

- EVALUATION OF THE STRUCTURE AND WORKINGS OF AN ACTUAL WORKING SYSTEM
- DISCUSSION OF CODE/STRUCTURE CHANGES REQUIRED FOR SYSTEM MODIFICATIONS
- WALKTHROUGH OF MODIFIED SYSTEM

SPECIAL CONSIDERATIONS:

- THIS SECTION PROVIDES DIRECT EXPERIENCE AS THE MOTIVATION FOR THE FORMAL DISCUSSION OF GOOD/BAD CHARACTERISTICS OF A DESIGN AS COVERED IN SECTION 18
- ENCOURAGE STUDENT INTERACTION BY BREAKING CLASS INTO GROUPS OF 3 OR 4
- INSTRUCTOR MUST HAVE AN EXCELLENT GRASP OF THIS SECTION!

INSTRUCTOR NOTES

SECTION 17 - (Continued) - STUDENT PROBLEM

SUMMARY OF MAIN POINTS COVERED:

- STUDENTS EXPERIENCE FIRST-HAND DESIGN IN Ada

MAIN MESSAGES:

- MANAGERS ARE REQUIRED TO EVALUATE AN ACTUAL SYSTEM AND DECIDE WHAT THEY LIKE/DISLIKE ABOUT THE SYSTEM

SPECIAL CONSIDERATIONS:

- IT IS ONE THING TO LISTEN TO AN INSTRUCTOR TALK ABOUT WHAT SHOULD OR SHOULD NOT BE IN A "GOOD" Ada SYSTEM. IT IS QUITE ANOTHER TO TRY TO DO IT YOURSELF AND THEN DISCUSS THE STRONG AND WEAK POINTS. ONE TENDS TO LEARN BETTER FROM EXPERIENCE.
- STUDENTS SHOULD WORK IN GROUPS OF 3 OR 4 TO COME UP WITH COMMENTS ABOUT THE SYSTEM
- ASKING STUDENTS "WHAT HAPPENS IF ..." QUESTIONS REQUIRES THAT THEY ACTUALLY SPEND TIME UNDERSTANDING THE SYSTEM

INSTRUCTOR NOTES

SECTION 17 - (Continued) - DISCUSSION

SUMMARY OF MAIN POINTS COVERED:

- DEVELOP A CONCEPTUAL (INTUITIVE) UNDERSTANDING OF DESIGN/CODE CHARACTERISTICS TO BE AWARE OF

MAIN MESSAGES:

- IT IS EASY TO HAVE GOOD-LOOKING Ada CODE, BUT HARDER TO KNOW IF IT IS BASED ON A GOOD DESIGN

SPECIAL CONSIDERATIONS:

- STRESS MAINTAINABILITY, RELIABILITY, UNDERSTANDABILITY
- MAKE SURE THE STUDENTS ANSWER THE QUESTIONS ABOUT THE SYSTEM
    - "HOW DOES IT WORK? QUESTIONS
    - "HOW CAN IT BE IMPROVED?" QUESTIONS
    - "HOW EASILY CAN WE CHANGE IT? QUESTIONS

INSTRUCTOR NOTES

SECTION 17 - (Continued) - REVISED SOLUTION

SUMMARY OF MAIN POINTS COVERED:

- DEPICTS A SOLUTION THAT SUPPORTS MORE OF THE SOFTWARE ENGINEERING GOALS

MAIN MESSAGES:

- BY DESIGNING/CODING WITH THE SOFTWARE ENGINEERING GOALS/PRINCIPLES IN MIND WE CAN ACHIEVE A SYSTEM THAT IS EASIER TO UNDERSTAND AND MODIFY

SPECIAL CONSIDERATIONS:

- EMPHASIZE HOW MUCH EASIER THE REVISED SOLUTION IS TO UNDERSTAND AND MODIFY

INSTRUCTOR NOTES

SECTION 18 - CHARACTERISTICS OF GOOD Ada DESIGNS

SUMMARY OF MAIN POINTS COVERED:

- FORMALIZE THE INTUITIVE CHARACTERISTICS OF A GOOD Ada DESIGN FROM THE STUDENT EXAMPLE OF SECTION 17

MAIN MESSAGES:

- WHEN EVALUATING AN Ada DESIGN/CODE THINK OF THE SOFTWARE ENGINEERING GOALS: UNDERSTANDABILITY, MODIFIABILITY, RELIABILITY, EFFICIENCY AND REUSABILITY

SPECIAL CONSIDERATIONS:

- GENERAL Ada GUIDE FOR MANAGER EVALUATION OF DESIGN/CODE IS GIVEN
- GUIDE INCLUDES
  - IS DESIGN/CODE EASY TO UNDERSTAND?
  - IS IT EASY TO GET AN OVERALL "BIG PICTURE"?
  - DOES IT SEEM NATURAL?
- EMPHASIZE THAT THESE ARE THE QUESTIONS THIS MODULE HAS BEEN PREPARING THEM TO ANSWER

INSTRUCTOR NOTES

- REUSABILITY WILL DECREASE SOFTWARE COSTS. INSTRUCTORS IN TRAINING SHOULD EMPHASIZE THIS.

- PORTABILITY OF SOFTWARE SEEKS TO MINIMIZE CHANGES TO SOFTWARE. TOTAL PORTABILITY IS NOT GENERALLY POSSIBLE, SO INSTRUCTORS IN TRAINING SHOULD BE CAREFUL NOT TO GIVE THE IMPRESSION THAT WE ARE SAYING IT IS.

SECTION 19 - Ada IN PERSPECTIVE : REUSABILITY AND PORTABILITY

SUMMARY OF MAIN POINTS COVERED:

- Ada FEATURES THAT ENCOURAGE REUSABILITY
- Ada FEATURES THAT INHIBIT PORTABILITY

MAIN MESSAGES:

- REUSABILITY REQUIRES PLANNING
- PORTABILITY REQUIRES PLANNING

SUBTOPICS:

- Ada FEATURES AND REUSABILITY
- CHARACTERISTICS OF PORTABLE SOFTWARE
- Ada FEATURES AND PORTABILITY

SPECIAL CONSIDERATIONS:

- EMPHASIZE THAT REUSABILITY AND PORTABILITY ARE DESIRABLE, REACHABLE GOALS
  - REQUIRE PLANNING AND EXPERIENCE
- PORTABILITY IS NOT ABSOLUTE
  - IF DEVICES ON ONE SYSTEM ARE COMPLETELY FROM DEVICES ON ANOTHER, THEN DEVICE HANDLERS MUST BE REWRITTEN, BUT USE OF DEVICE HANDLERS CAN PROBABLY STAY THE SAME
  - GOAL OF PORTABILITY IS TO DECREASE REQUIRED CHANGES (ONE MONTH EFFORT RATHER THAN ONE YEAR)

INSTRUCTOR NOTES

- ALLOW 30 MINUTES FOR THIS SECTION

SECTION 5

L303

REAL TIME CONCEPTS

VG 931/B

INSTRUCTOR NOTES

- EMPHASIZE TO THE INSTRUCTORS IN TRAINING THAT THEY SHOULD FIND OUT ABOUT THE CONCURRENT/REAL TIME PROGRAMMING BACKGROUND OF THE STUDENTS
  - FOR A CLASS WITH A STRONG BACKGROUND, LESS EMPHASIS CAN BE PLACED ON THE INTRODUCTORY MATERIAL
  - FOR A CLASS WITH A WEAK BACKGROUND, EMPHASIZE INTRODUCTORY MATERIAL AND CONCEPTS BUT PLACE LESS EMPHASIS ON DETAILS OF SPECIFIC TASK DESIGNS

  SUGGESTIONS FOR EMPHASIS APPEAR THROUGHOUT THE NOTES

GENERAL COMMENTS

- THERE ARE NO IN-CLASS EXERCISES OR LAB EXERCISES FOR THIS MODULE
- SPECIAL CONSIDERATIONS ARE NOTED WHERE APPLICABLE, BASED ON PRIOR EXPERIENCE TEACHING THE MATERIAL
- EACH SECTION IS DISCUSSED, ITS MAIN POINTS AND ITS RELATION TO THE COURSE OVERALL
- IT IS IMPORTANT TO REMEMBER WHEN TEACHING THIS SECTION THAT THE STUDENTS ARE <u>NOT</u> ASSUMED TO HAVE ANY CONCURRENT PROGRAMMING OR REAL TIME PROGRAMMING BACKGROUND

INSTRUCTOR NOTES

- GIVE THE INSTRUCTORS IN TRAINING AN OVERVIEW OF WHAT IS COVERED IN L303 AND WHAT ITS OBJECTIVES ARE.

- TARGET TEACHING TIME IS IN PARENTHESES
  - INCLUDED TO GIVE INSTRUCTORS IN TRAINING AN IDEA OF HOW MUCH TIME IS DEVOTED TO VARIOUS TOPICS
  - EMPHASIZE THAT TIME IS TARGET. MAY VARY DEPENDING ON CLASS NEEDS/BACKGROUND.

OVERVIEW

SECTION 1 - CONCURRENT PROGRAMMING CONCEPTS (1:30)

- GIVES AN OVERVIEW OF CONCURRENT PROGRAMMING AND ITS PROBLEMS

SECTION 2 - Ada TASKING FEATURES (2:00)

- DESCRIBES Ada FEATURES AVAILABLE FOR TASKING

SECTION 3 - FUNDAMENTAL TASK DESIGNS (1:30)

- DESCRIBES WAYS TASKS CAN BE COMBINED

SECTION 4 - IMPROVING PERFORMANCE (:30)

- DISCUSSES PERFORMANCE ISSUES

INSTRUCTOR NOTES

SECTION 1 - CONCURRENT PROGRAMMING CONCEPTS

SUMMARY OF MAIN POINTS COVERED:

- BASIC CONCEPTS OF CONCURRENT PROGRAMMING
- REASONS FOR WRITING CONCURRENT PROGRAMS
- COMMON PITFALLS OF CONCURRENT PROGRAMMING

MAIN MESSAGES:

- WE ARE USED TO THINKING OF A PROGRAM'S EXECUTION AS CONSISTING OF A SEQUENCE OF ACTIONS. HOWEVER, A PROGRAM MAY CONSIST OF MORE THAN ONE SEQUENCE OF ACTIONS.
- CONCURRENCY PROVIDES A MORE NATURAL WAY OF LOOKING AT CERTAIN PROBLEMS, INCLUDING SOME PROBLEMS WE NORMALLY THINK OF AS SEQUENTIAL
- CONCURRENT PROGRAMMING *INTRODUCES PROBLEMS NOT PRESENT* IN SEQUENTIAL PROGRAMMING

SUBTOPICS:

- DEFINITION OF A PROCESS
- OVERLAPPED VERSUS INTERLEAVED CONCURRENCY
- ASYNCHRONOUS PROCESSES
- RE-ENTRANCY
- ROLE OF A RUNTIME SYSTEM
- MANAGEMENT OF SIMULTANEOUS REAL TIME ACTIVITIES
- SIMULATION OF SIMULTANEOUS ACTIVITIES
- LOGICAL DECOMPOSITION OF A PROBLEM INTO STREAM TRANSFORMATIONS
- SIMULTANEOUS UPDATE
- DEADLOCK
- STARVATION
- COOPERATION AMONG PROCESSES

INSTRUCTOR NOTES

- THE MATERIAL IN THIS SECTION IS INTRODUCTORY.
  - FOR STUDENTS WITH A WEAK BACKGROUND, THIS PROVIDES A GOOD DISCUSSION OF THE USES OF CONCURRENT/REAL TIME PROGRAMMING AND ITS PROBLEMS. IF THE CLASS HAS A WEAK BACKGROUND, GO THROUGH THIS MATERIAL AT A SLOWER PACE
  - FOR STUDENTS WITH A STRONG BACKGROUND, THIS WILL BE OLD HAT. FOR A CLASS WITH A STRONG BACKGROUND, GO THROUGH THIS MATERIAL AT A FASTER PACE. THE EMPHASIS SHOULD BE ON MAKING STUDENTS AWARE THAT Ada CAN BE USED TO SOLVE TASKING PROBLEMS.

- WHEN DISCUSSING RUNTIME SYSTEMS, MAKE SURE CLASS UNDERSTANDS DIFFERENT RUNTIME SYSTEMS WILL EXIST.

SPECIAL CONSIDERATIONS:

- FOR SOME STUDENTS, THIS MIGHT BE THEIR FIRST EXPOSURE TO CONCURRENT PROGRAMMING
  - USUALLY THINK OF PROGRAM EXECUTION AS CONSISTING OF SINGLE SEQUENCE OF ACTIONS
  - MIGHT HAVE DIFFICULTY UNDERSTANDING HOW A PROGRAM CAN CONSIST OF MORE THAN ONE SEQUENCE OF ACTIONS
- DISCUSSION OF CONCURRENT PROCESS SHOULD BE KEPT ALMOST ENTIRELY LANGUAGE INDEPENDENT
  - PROCESSES ARE INDEPENDENT OF ANY PROGRAMMING LANGUAGE
  - PROCESS IS NOT SYNONYMOUS WITH Ada TASK
  - MAKE SURE YOU USE "PROCESS" NOT "TASK"
  - ROLE OF RUNTIME SYSTEM WILL BE OF SPECIAL INTEREST TO MANY STUDENTS SO MAKE SURE THEY UNDERSTAND
    - RUNTIME SYSTEM FOR ONE APPLICATION MAY NOT BE APPROPRIATE FOR ANOTHER, E.G., AN ALS RUNTIME SYSTEM WOULD NOT BE USED ON AN F-14
    - EVENTUALLY WILL HAVE OFF-THE-SHELF RUNTIME SYSTEMS
    - Ada RUNTIME SYSTEMS CAN BE ADAPTED TO MEET APPLICATION NEEDS AS LONG AS THEY DON'T VIOLATE Ada RULES
  - WHEN DISCUSSING REASONS FOR WRITING CONCURRENT PROGRAMS, EMPHASIZE COMMON THEMES (AND AVOID DETAILED DISCUSSIONS OF THE EXAMPLES)
  - THE MATERIAL ON COMMON PITFALLS SHOULD BE KEPT LIGHT
    - ADD ANY ANECDOTES YOU ARE AWARE OF
    - ENCOURAGE STUDENTS TO DO THE SAME

INSTRUCTOR NOTES

- THIS SECTION DESCRIBES BASIC FACTS ABOUT Ada TASKING - TASK TYPES, TASK OBJECTS, RENDEZVOUS, SELECT STATEMENTS - AND SURVEYS OTHER TASK FEATURES - EXCEPTIONS IN TASKS, PRIORITIES, INTERRUPT ENTRIES, ETC.

SECTION 2 - Ada TASKING FEATURES

SUMMARY OF MAIN POINTS COVERED:

- TASK TYPES AND TASK OBJECTS AND HOW THEY ARE DECLARED, ACTIVATED, AND TERMINATED
- RENDEZVOUS AND STATEMENTS FOR CONTROLLING THEM
- OTHER TASKING FEATURES

MAIN MESSAGES:

- TASKS ARE DATA OBJECTS BELONGING TO TASK TYPES
- LIKE OTHER PROGRAM UNITS, TASK TYPES ARE DEFINED BY A DECLARATION DESCRIBING THE INTERFACE AND A BODY DESCRIBING THE IMPLEMENTATION
- TASK OBJECTS BEGIN PARALLEL EXECUTION AT ABOUT THE TIME THEY ARE CREATED, AND DEPARTURE FROM CERTAIN SEQUENCES OF STATEMENTS DOES NOT OCCUR UNTIL CERTAIN TASK OBJECTS HAVE TERMINATED
- TASKS COMMUNICATE WHEN AN ACCEPT STATEMENT ACCEPTS AN ENTRY CALL
- TASKS CAN EXERCISE VARYING DEGREES OF CONTROL OVER ENTRY CALLS THEY MAKE AND ACCEPT, THE CONDITIONS UNDER WHICH THEY ACCEPT ENTRY CALLS, AND THE AMOUNT OF TIME THEY WAIT FOR RENDEZVOUS
- SPECIAL RULES GOVERN RAISING OF EXCEPTIONS DURING TASK COMMUNICATION AND THE PROPAGATION OF EXCEPTIONS FROM ONE TASK TO ANOTHER
- ONE TASK CAN TERMINATE ANOTHER BY USING THE abort STATEMENT BUT THERE ARE USUALLY MORE APPROPRIATE WAYS TO GET A TASK TO TERMINATE
- A HARDWARE INTERRUPT CAN BE MADE TO LOOK LIKE AN ENTRY CALL
- PRIORITIES MAY BE ASSIGNED TO TASKS TO INDICATE RELATIVE DEGREES OF URGENCY, AND RESOLVE CONTENTION FOR THE USE OF A CPU, BUT NOT TO SYNCHRONIZE TASKS

INSTRUCTOR NOTES

- MAKE SURE INSTRUCTORS IN TRAINING REALIZE IMPORTANCE OF USING THE TERM "TASK".

- EMPHASIZE TO THE INSTRUCTORS IN TRAINING THAT IF THEY DO NOT DISTINGUISH BETWEEN "TASK INITIATION" AND A "TASK BEING INITIATED FOR ITS NEXT DUTY CYCLE", CONFUSION MAY WELL OCCUR.

SUBTOPICS:

- TASKS AS DATA OBJECTS
- TASK DECLARATIONS AND TASK BODIES
- OVERVIEW OF TASK INITIATION AND TERMINATION
- SIMPLE RENDEZVOUS
- SELECTIVE WAITS
- TIMED AND CONDITIONAL ENTRY CALLS
- EXCEPTIONS IN TASKS
- THE ABORT STATEMENT
- INTERRUPT ENTRIES
- ENTRY FAMILIES
- TASK PRIORITIES

SPECIAL CONSIDERATIONS:

- START USING "TASK" RATHER THAN "PROCESS"
- MAKE SURE CLASS UNDERSTANDS TASK TYPES MAY APPEAR IN SIMPLE VARIABLES, ARRAY OR RECORD COMPONENTS, AND ACCESS TYPE COMPONENTS
- MAKE SURE CLASS THINKS OF TASK OBJECTS AS DATA OBJECTS
- STUDENTS FAMILIAR WITH CYCLIC EXECUTIVES MAY CONFUSE TASK INITIATION WITH A TASK BEING "INITIATED" FOR ITS NEXT DUTY CYCLE AND THEN TERMINATING UNTIL ITS NEXT TURN
  - IF THE CLASS CONTAINS MANY EXPERIENCED REAL TIME PROGRAMMERS, THEN EMPHASIZE DISTINCTION; OTHERWISE, DO NOT SPEND MUCH TIME ON THIS POINT
- THE MATERIAL ON OTHER TASKING FEATURES SUCH AS EXCEPTIONS, INTERRUPT ENTRIES, ETC., IS PRESENTED AT A VERY HIGH LEVEL
  - NOT GIVING DETAILS SINCE STUDENTS ARE NOT EXPECTED TO BE CODING
  - SIMPLY LETTING STUDENTS KNOW THESE FEATURES EXIST

INSTRUCTOR NOTES

- THIS SECTION COVERS A WIDE RANGE OF FUNDAMENTAL TASK DESIGNS
  - SOME, LIKE MONITORS AND MESSAGE BUFFERS, CAN BE USED AS BUILDING BLOCKS
  - CYCLIC PROCESSING SHOWS Ada SOLUTIONS TO STANDARD REAL TIME PROGRAMMING PROBLEMS
  - STREAM-ORIENTED TASK DESIGN SHOWS HOW CONCURRENT PROGRAMMING CAN BE USED EFFECTIVELY EVEN WHEN NOT IMMEDIATELY CALLED FOR BY THE PROBLEM

SECTION 3 - FUNDAMENTAL TASK DESIGNS

SUMMARY OF MAIN POINTS COVERED:

- ASYMMETRY OF RENDEZVOUS
- USING TASKS TO MANAGE ACCESS TO SHARED DATA
- USING AND IMPLEMENTING MESSAGE BUFFERS
- STREAM-ORIENTED TASK DESIGN
- CYCLIC PROCESSING

MAIN MESSAGES:

- THE UNDERLYING VIEWPOINT ON WHICH Ada TASKING DESIGNS ARE BASED:
  - ONE TASK CALLS ANOTHER TO OBTAIN SOME SERVICE FROM THAT TASK
  - A TASK ACCEPTS AN ENTRY CALL IN ORDER TO PERFORM A SERVICE FOR THE CALLING TASK
  - A CALLED TASK "KNOWS" WHOM IT IS CALLING, BUT A CALLED TASK DOES NOT "KNOW" WHO IS CALLING IT
- A MONITOR TASK ALLOWS SEVERAL OTHER TASKS TO SHARE DATA WITHOUT INTERFERING WITH EACH OTHER
- JUST AS A PACKAGE PROVIDES A LIMITED NUMBER OF OPERATIONS THROUGH WHICH A SINGLE TASK CAN MANIPULATE DATA HIDDEN INSIDE THE PACKAGE, A MONITOR TASK PROVIDES A LIMITED NUMBER OF OPERATIONS THROUGH WHICH SEVERAL CONCURRENT TASKS CAN MANIPULATE DATA HIDDEN INSIDE THE MONITOR
- IT IS NATURAL TO THINK OF A MONITOR AS A DATA OBJECT
- MESSAGE BUFFERS ALLOW TASKS TO COMMUNICATE WITHOUT WAITING FOR EACH OTHER
- RENDEZVOUS CAN BE USED AS PRIMITIVES TO BUILD MORE ELABORATE MECHANISMS FOR TASK COMMUNICATION
- TASKS CAN BE USED TO SOLVE PROBLEMS THAT ARE NOT INHERENTLY CONCURRENT
- TRADITIONAL CYCLIC PROCESSING CAN BE PERFORMED IN Ada
- Ada PROVIDES ALTERNATIVES TO TRADITIONAL CYCLIC PROCESSING

INSTRUCTOR NOTES

- MAKE SURE THE INSTRUCTORS IN TRAINING EMPHASIZE THE BUILDING BLOCK VIEW OF MONITORS AND MESSAGE BUFFERS
- FOR STUDENTS WITH A WEAK BACKGROUND IN REAL TIME PROGRAMMING, JUST GIVE A QUICK OVERVIEW OF CYCLIC PROCESSING

SUBTOPICS:

- SERVER AND USER TASKS
- SIMULTANEOUS UPDATE
- MONITORS AS DATA ABSTRACTION TOOLS
- TASK COMMUNICATION WITHOUT BUFFERING
- ONE-ELEMENT MESSAGE BUFFER
- N-ELEMENT MESSAGE BUFFER
- UNBOUNDED MESSAGE BUFFER
- STRUCTURE CLASH
- STREAM TRANSFORMATIONS
- CO-ROUTINES
- CYCLIC EXECUTIVES
- CUMULATIVE DRIFT
- JITTER

SPECIAL CONSIDERATIONS:

- EMPHASIZE DESIGN HINT: SOMETIMES THE DESIGN OF A MULTI TASK PROGRAM CAN BE SIMPLIFIED BY REVERSING THE DIRECTION OF RENDEZVOUS BETWEEN TWO TASKS
- EMPHASIZE THE USE OF MONITORS FOR SAFELY SHARING DATA
- MENTION THAT MESSAGE BUFFERS ARE ALSO CALLED MESSAGE QUEUES, BUT USE THE TERM MESSAGE BUFFER
- WHEN DISCUSSING IMPLEMENTING MESSAGE BUFFER IN HARDWARE, MAKE SURE STUDENTS DO NOT THINK YOU ARE SAYING TASKING IS INEFFICIENT
- REMEMBER WHEN DISCUSSING CYCLIC PROCESSING THAT REAL TIME/CONCURRENT PROGRAMMING IS <u>NOT</u> A PREREQUISITE FOR THIS MODULE
- EMPHASIZE THAT $\overline{\text{A T}}$RADITIONAL CYCLIC EXECUTIVE CAN BE WRITTEN IN Ada BUT THAT A MULTI-THREAD APPROACH SHOULD ALSO BE CONSIDERED
- STREAM-ORIENTED IS A VIABLE DESIGN TECHNIQUE EVEN IF DESIGN MUST BE REALIZED AS A SINGLE TASK: DETAILS NOT PRESENTED IN THIS MODULE (BUT THEY ARE IN L401)

INSTRUCTOR NOTES

- INSTRUCTORS IN TRAINING NEED TO REALIZE THAT WHEN TEACHING L303, STUDENTS MUST REALIZE
  - WE APPRECIATE THEIR CONCERNS FOR EFFICIENCY
  - CARE MUST BE TAKEN WHEN TUNING. THIS SECTION DISCUSSES THE PROBLEMS OF PREMATURE AND MISGUIDED TUNING
  - THE TOPIC OF PERFORMANCE IS COVERED IN MUCH GREATER DETAIL IN L401
- THE QUOTE ON KEEPING THE PROPER PERSPECTIVE SIMPLY REMINDS MANAGERS THAT STRUCTURED PROGRAMMING AND EFFICIENCY ARE <u>BOTH</u> IMPORTANT

SECTION 4 - IMPROVING PERFORMANCE

SUMMARY OF MAIN POINTS COVERED:

- TUNING TO MEET REAL TIME CONSTRAINTS
- PREMATURE TUNING

MAIN MESSAGES:

- MAKE SURE YOU REALLY HAVE A PERFORMANCE PROBLEM BEFORE TUNING IS STARTED
- ONLY SMALL PORTIONS OF A PROGRAM NEED BE TUNED
- PREMATURE TUNING OFTEN RESULTS IN WASTED EFFORT AND INCREASED PROGRAM COMPLEXITY

SUBTOPICS:

- HOW MUCH OF A PROGRAM SHOULD BE TUNED
- PROFILES
- PREMATURE TUNING

SPECIAL CONSIDERATIONS:

- KEEP THIS SECTION LIGHT
- ENCOURAGE STUDENTS TO SHARE THEIR EXPERIENCES WITH TUNING
- EMPHASIZE THAT TUNING SHOULD <u>ONLY</u> BE PERFORMED ONCE THE DESIGN HAS BEEN PROPERLY (CORRECTLY AND UNDERSTANDABLY) IMPLEMENTED
- MAKE SURE STUDENTS REALIZE THAT THEIR CONCERNS WITH EFFICIENCY ARE UNDERSTOOD
  - THIS SHOULD BE EMPHASIZED RIGHT AT THE START
  - EMPHASIZE IT ONE MORE TIME AT THE END WITH THE BENTLEY QUOTE ON KEEPING THE PROPER PERSPECTIVE

INSTRUCTOR NOTES

SUGGESTED READING FOR L303 INSTRUCTORS

- THE "BUG" HEARD 'ROUND THE WORLD, JOHN R. GORMAN, ACM, SIGSOFT SOFTWARE ENGINEERING NOTES, VOL. 6, NO. 5, OCTOBER 1981, 3-10
  - DESCRIBES SOFTWARE BUG THAT POSTPONED FIRST FLIGHT OF THE SPACE SHUTTLE
  - SHOWS HOW DIFFICULT TASK SYNCHRONIZATION
  - BACKGROUND READING ONLY

- EVOLVING TOWARD Ada IN REAL TIME SYSTEMS, LEE MACLAREN, ACM SIGPLAN NOTICES, VOL. 15, NO. 11, NOVEMBER 1980, 146-155
  - DISCUSSES CYCLIC EXECUTIVES AND THEIR PROBLEMS
  - IMPLEMENTATION IN Ada
  - SHOULD READ THIS PAPER BEFORE TEACHING SECTION ON CYCLIC PROCESSING

- PROGRAMMING PEARLS, JON L. BENTLEY, COMMUNICATIONS OF THE ACM, BIMONTHLY FEATURE
  - WIDE RANGE OF PROGRAMMING TOPICS DISCUSSED
  - FREQUENTLY DISCUSSES EFFICIENT PROGRAMMING TECHNIQUES
  - SOURCE OF ADDITIONAL WAR STORIES
  - SHOULD READ REGULARLY

- WRITING EFFICIENT PROGRAMS, JON L. BENTLEY, PRENTICE-HALL, 1982
  - EXCELLENT SOURCE FOR WRITING EFFICIENT PROGRAMS
  - DISCUSSES TUNING TECHNIQUES

- L401 - REAL TIME SYSTEMS IN Ada
  - COVERS L303 TOPICS IN GREATER DETAIL